Outline 2017/05/05

- Data analysis and data selection
- TCanvas::Divide(), TGraphAsymmErrors as TH1 ratio, ...
- THN with arbitrary binning
- TTree::MakeClass()
- How to compile a ROOTable library
- ROOT 5 vs ROOT 6
- Hands on software
  - make scatter plots
  - read a tree with class using a script
  - fitting distributions
  - look for e+ in PAMELA simulation

Outline 2017/05/05

2017 MAY

SUN	MON	TUE	WED	THU	FRI	SAT
	1	2	3	4	5 OK	6
7	8	9	10	11	12 OK	13
14	15	16	17	18	19 OK	20
21	22 9-13 ?	23 14-17 ?	24	25	26 OK	27
28	29	30	31			

Monday June 5th 9-13 ?

www.free-printable-calendar.com

- ❑ May 22nd 9-13
- ❑ May 23rd 14-17
- ❑ June 5th 9-13

Fitting

Distributions, profiles, graph can always be fitted with the method called “Fit”.

fit options

```
TFitResultPtr Fit(const char* formula, Option_t* option = "", Option_t*
   goption = "", Double_t xmin = 0, Double_t xmax = 0)

TFitResultPtr Fit(TF1* f1, Option_t* option = "", Option_t* goption
   = "", Double_t xmin = 0, Double_t xmax = 0)
```

graphic options

formula: “[0] + x*[1] + x*x*[2]” == “pol2” that is: a + x*b + x^2*c

TF1* f1:

```
TF1 *f=new TF1(“f”, “[0] + x*[1] + x*x*[2]”,0.,100.)
```

http://root.cern.ch/root/html532/TH1.html#TH1:Fit

http://root.cern.ch/root/html532/TGraph.html#TGraph:Fit

http://root.cern.ch/root/html532/TF1.html

Fitting, predefined functions

Distributions, profiles, graph can always be fitted with the method called “Fit”.

fit options

```
TFitResultPtr Fit(const char* formula, Option_t* option = "", Option_t*
   goption = "", Double_t xmin = 0, Double_t xmax = 0)

TFitResultPtr Fit(TF1* f1, Option_t* option = "", Option_t* goption
   = "", Double_t xmin = 0, Double_t xmax = 0)
```

graphic options

formula: “gaus” , “expo” , “landau” , “pol*N*” (i.e.: “pol0”, “pol1”, “pol2”, ...)

http://root.cern.ch/root/html532/TH1.html#TH1:Fit

http://root.cern.ch/root/html532/TGraph.html#TGraph:Fit

http://root.cern.ch/root/html532/TF1.html

Fitting, setting initial conditions

Setting initial conditions is easier when using a function. It is done like using the TFormula::SetParameters method:

```
TF1 *pointerfun = new TF1("fun","[0]+x*[1]/2334.",1.,6.);

pointerfun->SetParameters(0.1,2000.);

mygraph->Fit(pointerfun, “me”, “same”);
```

Look also at the other methods like:

SetParLimits, FixParameter, etc.

Fitting, accessing the results (1/2)

The function returns a TFitResultPtr which can hold a pointer to a TFitResult object.

Use the fitting option "S" to make TFitResultPtr containing the TFitResult and behaves as a smart pointer to it. For example one can do:

```
TFitResultPtr r = h->Fit(“pol2”,“meS"); // NB: _NOT_ *r
Double_t chi2 = r->Chi2(); // to retrieve the fit chi2
Double_t par0 = r->Parameter(0); // retrieve the value for the
   parameter 0
Double_t err0 = r->ParError(0); // retrieve the error for the parameter
   0
r->Print("V"); // print full information of fit including covariance
   matrix
```

Fitting, accessing the results (2/2)

When using a function to perform the fit, results can be accesses via TFormula and TF1 methods GetParameter, GetParError, etc.:

```
TF1 *myFun = new TF1(“fun”,“pol2”);
h->Fit(myFun,“meS");
Double_t par0 = myFun->GetParameter(0); // retrieve the value for the
   parameter 0
Double_t err0 = myFun->GetParError(0); // retrieve the error for the
   parameter 0
```

Event selection

Today question is: where are positrons in our simulated sample?

Event selection goes trough a recursive procedure of plotting observables distributions and placing cut to select the events (standard procedure, other ways exist: multivariate approach, event selection “by eye”, ...)

In this case:

- electrons are representative of positrons, i.e. electrons and positrons distributions are identical, the only difference here is the sign of the charge of the particle (negative electrons, positive positrons).
- protons are the background we want to remove.

posSelection.C

```
#include <TFile.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TF1.h>
#include <TProfile.h>
#include <TGraphAsymmErrors.h>

#include <PamCalo.h>

void posSelection(TString inputFile){

  // Create some histograms.
  // electrons
  TH1D *elesample = new TH1D("elesample","Electron sample;GeV;counts",100,0.,20.);
  TH1D *elesel = new TH1D("elesel","Selected electrons;GeV;counts",100,0.,20.);
  // positrons
  TH1D *possel = new TH1D("possel","Selected positrons;GeV;counts",100,0.,20.);
  // sum ep + em
  TH1D *leptosel = new TH1D("leptpsel","Selected positrons+electrons;GeV;counts",100,0.,20.);

  // scatter plots
  TH2D *qtotene = new TH2D("qtotene","qtot/energy;GeV;qtot/energy",1000,-20.,20.,1000,0.,500.); // energy momentum match
  TH2D *noint = new TH2D("noint","noint;GeV;noint",1000,-20.,20.,1000,0.,35.); // noint distribution
  TH2D *qncore = new TH2D("qncore","qcore/ncore;GeV;qcore/ncore",1000,-20.,20.,5000,0.,50.); // qcore/ncore distribution

  // Functions for selections




  // Open the file
  TFile *file = TFile::Open(inputFile);
  TTree *tree = (TTree*)file->Get("pamcalotree");
  PamCalo *pc = new PamCalo();
  tree->SetBranchAddress("PamCalo",&pc);
```

posSelection.C

```
 tree->SetBranchAddress("PamCalo",&pc);

 // loop over events
 for ( Int_t i=0; i<tree->GetEntries(); i++) {
   //for ( Int_t i=0; i<20000; i++) {

   if ( i%10000 == 0 ) cout << " get entry " << i << "\n";

   tree->GetEntry(i);

   if ( pc->energy < 0. ) elesample->Fill(fabs(pc->energy));

   // selection condition (no selection at the moment)
//    if (
//       ){

     // fill scatter plots
     noint->Fill(pc->energy, pc->noint);
     if ( pc->ncore > 0 ) qncore->Fill(pc->energy,pc->qcore/(float)pc->ncore);
     if ( pc->energy != 0. ) qtotene->Fill(pc->energy,pc->qtot/fabs(pc->energy));
     // fill histrograms
     if ( pc->energy < 0. ){ // negatives particles
       elesel->Fill(fabs(pc->energy));
     } else { // positive particles
       possel->Fill(fabs(pc->energy));
     }
     leptosel->Fill(fabs(pc->energy));



 }
```

posSelection.C

```
// draw histograms
TCanvas *cqtotenergy = new TCanvas("cqtotenergy","Energy momentum match",200,10,600,400);
cqtotenergy->SetGrid();
cqtotenergy->SetTicks();
cqtotenergy->SetLogz();
cqtotenergy->Draw();
qtotene->Draw("colz");

TCanvas *cnoint = new TCanvas("cnoint","Distribution: noint",200,10,600,400);
cnoint->SetGrid();
cnoint->SetTicks();
cnoint->SetLogz();
cnoint->Draw();
noint->Draw("colz");


TCanvas *cqncore = new TCanvas("cqncore","Distribution: qcore/ncore",200,10,600,400);
cqncore->SetGrid();
cqncore->SetTicks();
cqncore->SetLogz();
cqncore->Draw();
qncore->Draw("colz");
```

ok for high statistics

replace with `hpointer->SetMarkerSize(1.4);`
`hpointer->Draw();`

when statistics becomes small

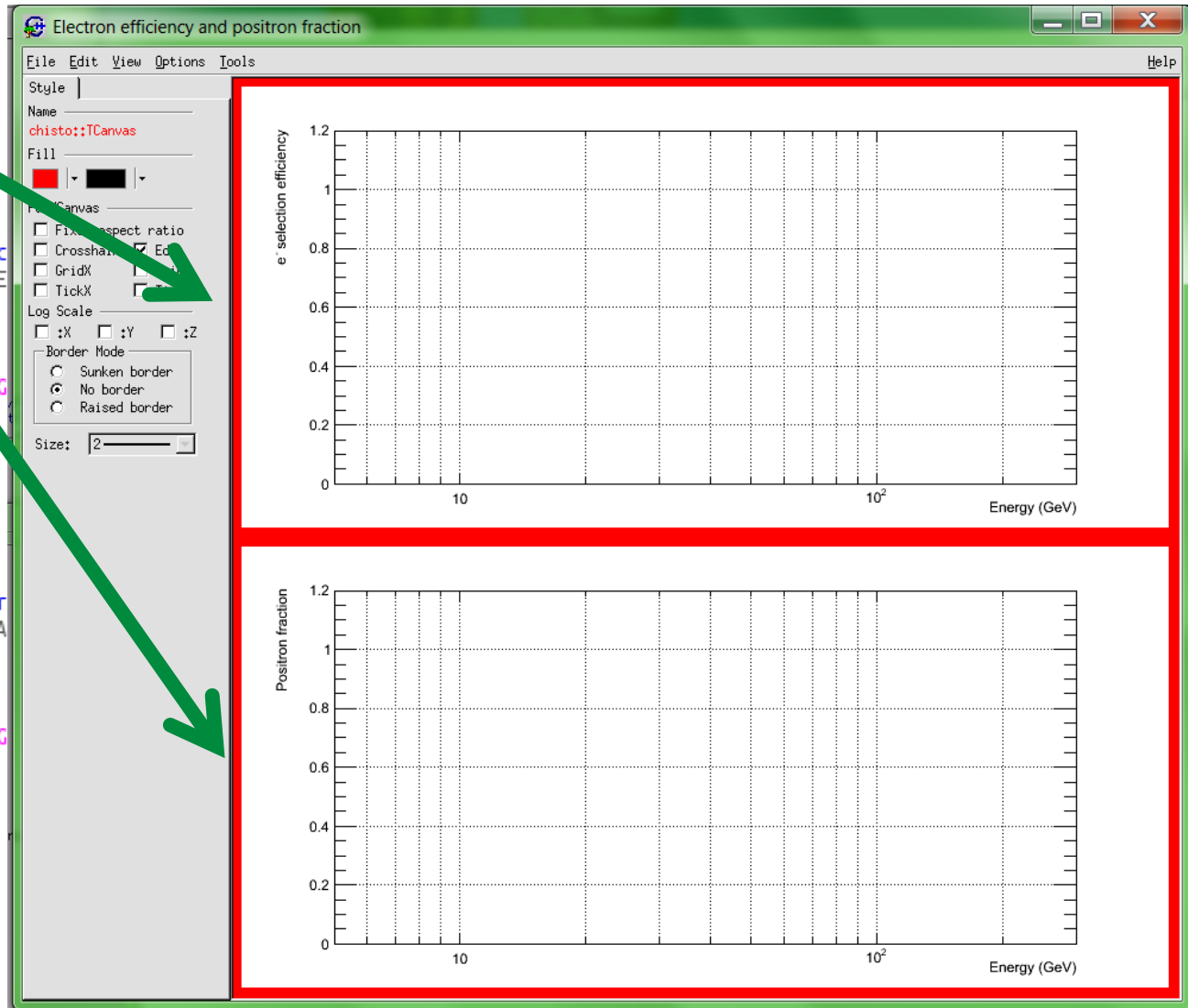
posSelection.C

```
// draw electron efficiency and positron fraction
TCanvas *chisto = new TCanvas("chisto","Electron efficiency and positron fraction",200,10,800,800);
chisto->Divide(1,2); // columns, raws
chisto->GetPad(1)->SetLogx();
chisto->GetPad(2)->SetLogx();
chisto->GetPad(1)->SetGrid();
chisto->GetPad(2)->SetGrid();
chisto->GetPad(1)->SetTicks();
chisto->GetPad(2)->SetTicks();
chisto->Draw();

// create electron efficiency, selected electrons / sample electrons
TGraphAsymmErrors* effele = new TGraphAsymmErrors(elesel,elesample);
effele->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgef = new TH2D("hbkgef",";Energy (GeV);e^{-} selection efficiency",1000,0.,30.,1000,0.,1.2);
hbkgef->SetStats(0);

// go to the first panel and draw
chisto->cd(1);
hbkgef->Draw();
effele->Draw("Psame");

// create positron fraction, selected positrons / (selected electrons + selected positrons)
TGraphAsymmErrors* posfraction = new TGraphAsymmErrors(possel,leptosel);
posfraction->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgpf = new TH2D("hbkgpf",";Energy (GeV);Positron fraction",1000,0.,30.,1000,0.,1.2);
hbkgpf->SetStats(0);

// go to the second panel and draw
chisto->cd(2);
hbkgpf->Draw();
posfraction->Draw("Psame");

}
```

posSelection.C

TPad::Divide(int,int)

```
// draw electron efficiency and positron fraction
TCanvas *chisto = new TCanvas("chisto","Electron efficiency and positron fraction",200,10,800,800);
chisto->Divide(1,2); // columns, raws
chisto->GetPad(1)->SetLogx();
chisto->GetPad(2)->SetLogx();
chisto->GetPad(1)->SetGrid();
chisto->GetPad(2)->SetGrid();
chisto->GetPad(1)->SetTicks();
chisto->GetPad(2)->SetTicks();
chisto->Draw();

// create electron efficiency, selected elec
TGraphAsymmErrors* effele = new TGraphAsymmE
effele->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgef = new TH2D("hbkgef",";Energy (G
hbkgef->SetStats(0);

// go to the first panel and draw
chisto->cd(1);
hbkgef->Draw();
effele->Draw("Psame");

// create positron fraction, selected positr
TGraphAsymmErrors* posfraction = new TGraphA
posfraction->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgpf = new TH2D("hbkgpf",";Energy (G
hbkgpf->SetStats(0);

// go to the second panel and draw
chisto->cd(2);
hbkgpf->Draw();
posfraction->Draw("Psame");

}
```



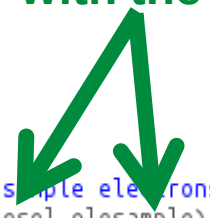
posSelection.C

```
// draw electron efficiency and positron fraction
TCanvas *chisto = new TCanvas("chisto","Electron efficiency and positron fraction",200,10,800,800);
chisto->Divide(1,2); // columns, raws
chisto->GetPad(1)->SetLogx();
chisto->GetPad(2)->SetLogx();
chisto->GetPad(1)->SetGrid();
chisto->GetPad(2)->SetGrid();
chisto->GetPad(1)->SetTicks();
chisto->GetPad(2)->SetTicks();
chisto->Draw();

// create electron efficiency, selected elec
TGraphAsymmErrors* effele = new TGraphAsymmE
effele->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgef = new TH2D("hbkgef",";Energy (G
hbkgef->SetStats(0);

// go to the first panel and dr
chisto->cd(1);
hbkgef->Draw();
effele->Draw("Psame");

// create positron fraction, selected positr
TGraphAsymmErrors* posfraction = new TGraphA
posfraction->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgpf = new TH2D("hbkgpf",";Energy (G
hbkgpf->SetStats(0);

// go to the second panel and draw
chisto->cd(2);
hbkgpf->Draw();
posfraction->Draw("Psame");

}
```

posSelection.C

```
// draw electron efficiency and positron fraction
TCanvas *chisto = new TCanvas("chisto","Electron efficiency and positron fraction",200,10,800,800);
chisto->Divide(1,2); // columns, raws
chisto->GetPad(1)->SetLogx();
chisto->GetPad(2)->SetLogx();
chisto->GetPad(1)->SetGrid();
chisto->GetPad(2)->SetGrid();
chisto->GetPad(1)->SetTicks();
chisto->GetPad(2)->SetTicks();
chisto->Draw();

// create electron efficiency, selected electrons / simple electrons
TGraphAsymmErrors* effele = new TGraphAsymmErrors(elesel,elesample);
effele->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgef = new TH2D("hbkgef",";Energy (GeV);e^{-} selection efficiency",1000,0.,30.,1000,0.,1.2);
hbkgef->SetStats(0);

// go to the first panel and draw
chisto->cd(1);
hbkgef->Draw();
effele->Draw("Psame");

// create positron fraction, selected positrons / (selected electrons + selected positrons)
TGraphAsymmErrors* posfraction = new TGraphAsymmErrors(possel,leptosel);
posfraction->SetMarkerStyle(20);

// create a TH2 to be used as background
TH2D *hbkgpf = new TH2D("hbkgpf",";Energy (GeV);Positron fraction",1000,0.,30.,1000,0.,1.2);
hbkgpf->SetStats(0);

// go to the second panel and draw
chisto->cd(2);
hbkgpf->Draw();
posfraction->Draw("Psame");

}
```

TH1D with the same binning!!

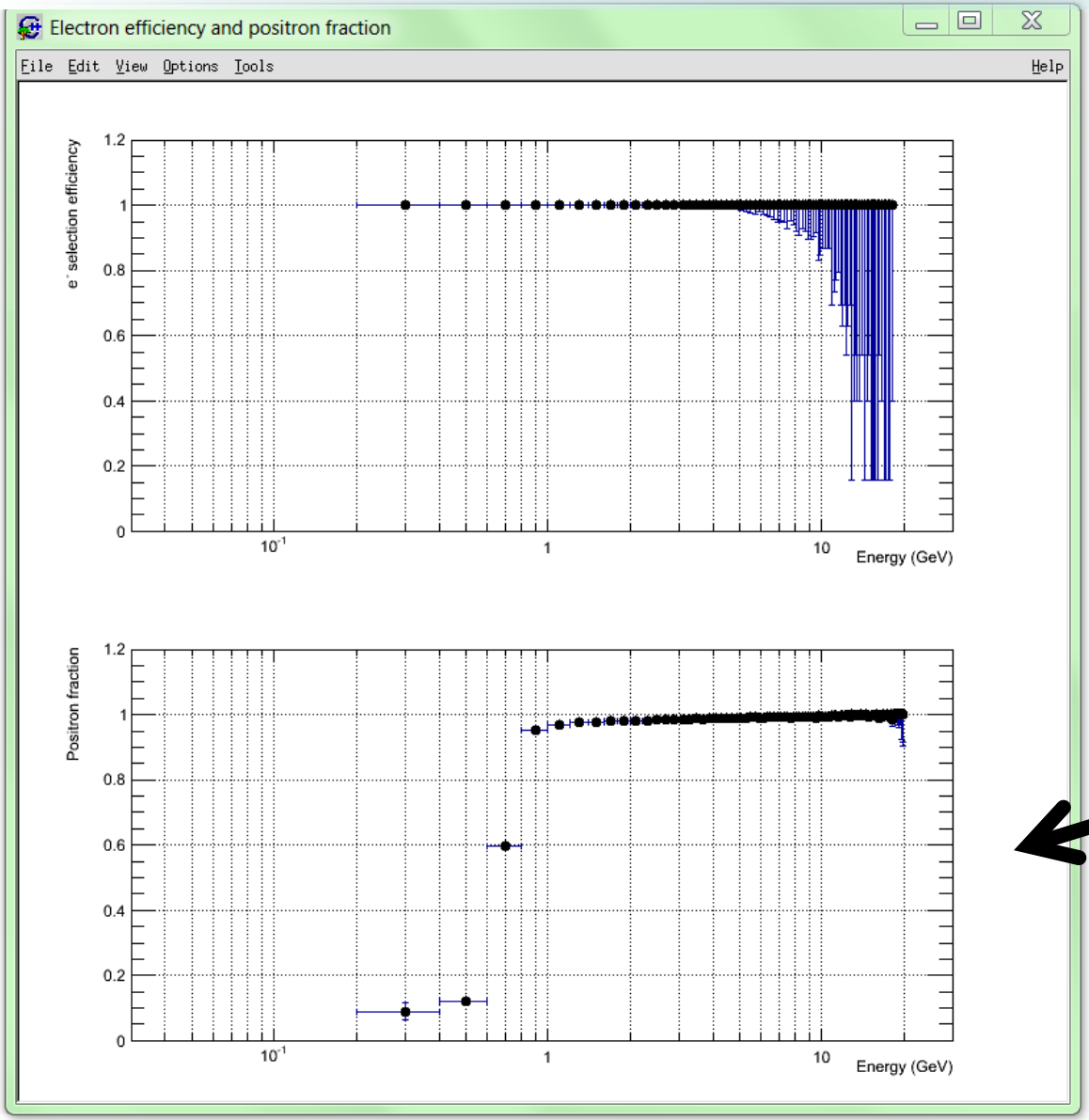


http://root.cern.ch/root/html532/TGraphAsymmErrors.html

Running the selection script

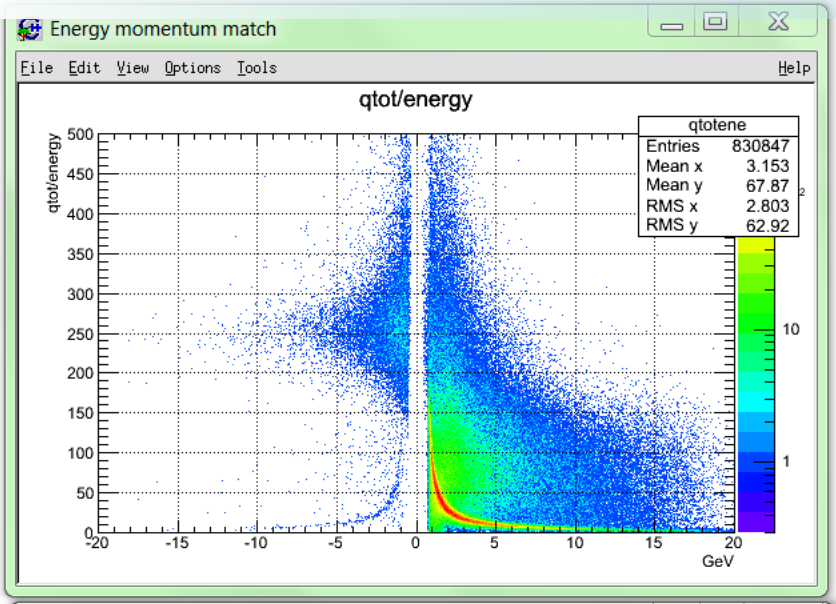
```
|Emi@marte ~>root
root [0] .L posSelection.C
root [1] posSelection(" /home/mocchiut/pamela/data/pamsimu2015.root")
 get entry 0
 get entry 10000
 get entry 20000
 get entry 30000
 get entry 40000
 get entry 50000
 get entry 60000
 get entry 70000
 get entry 80000
... snap ...
 get entry 5730000
```

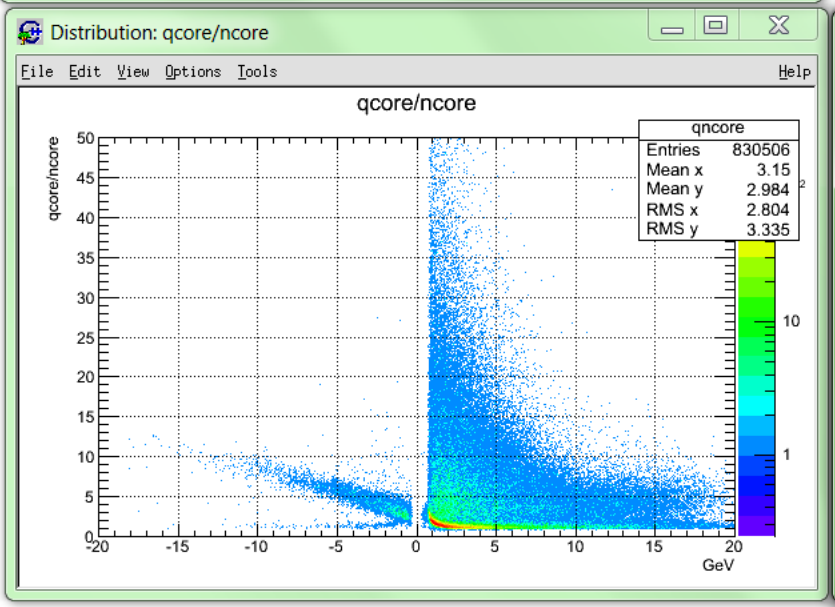
Output plots

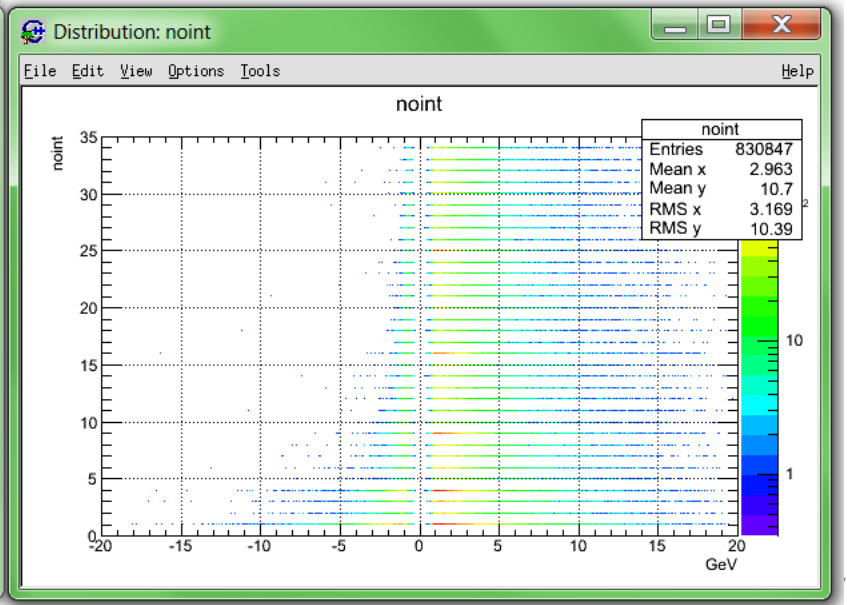


which value
do you expect here?

Output plots

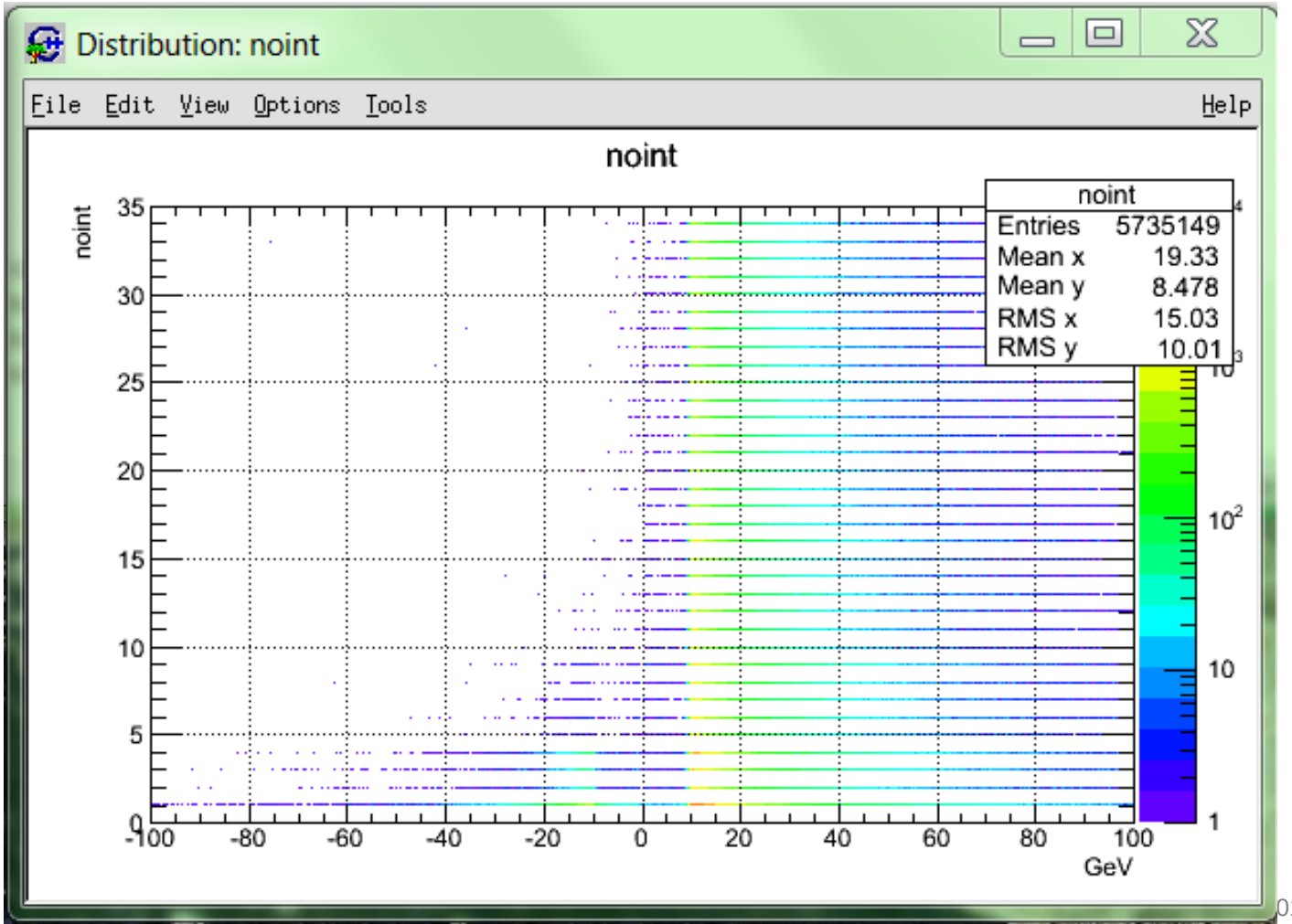






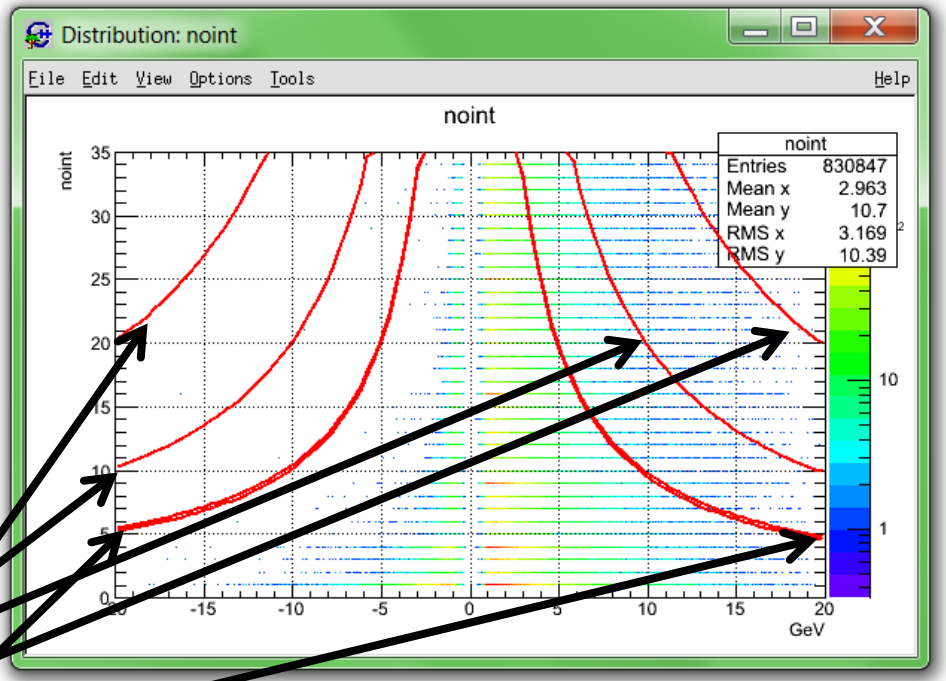
Choosing a cut, case one

Placing a cut using CINT and an individual empirical cut selection approach (by eye)



Cutting on “noint” distribution

```
 get entry 560000
 get entry 570000
 get entry 580000
 get entry 590000
 get entry 600000
 get entry 610000
 get entry 620000
 get entry 630000
 get entry 640000
 get entry 650000
 get entry 660000
 get entry 670000
 get entry 680000
 get entry 690000
 get entry 700000
 get entry 710000
 get entry 720000
 get entry 730000
 get entry 740000
 get entry 750000
 get entry 760000
 get entry 770000
 get entry 780000
 get entry 790000
 get entry 800000
 get entry 810000
 get entry 820000
 get entry 830000
root [2] TF1 *g = new TF1("g","abs(0.2+(-200./x))",-20,20)
root [3] g->Draw("same")
root [4] TF1 *g = new TF1("g","abs(0.2+(-400./x))",-20,20)
root [5] g->Draw("same")
root [6] TF1 *g = new TF1("g","abs(0.2+(-100./x))",-20,20)
root [7] g->Draw("same")
root [8] TF1 *g = new TF1("g","abs(0.5+(-100./x))",-20,20)
root [9] g->Draw("same")
root [10]
```



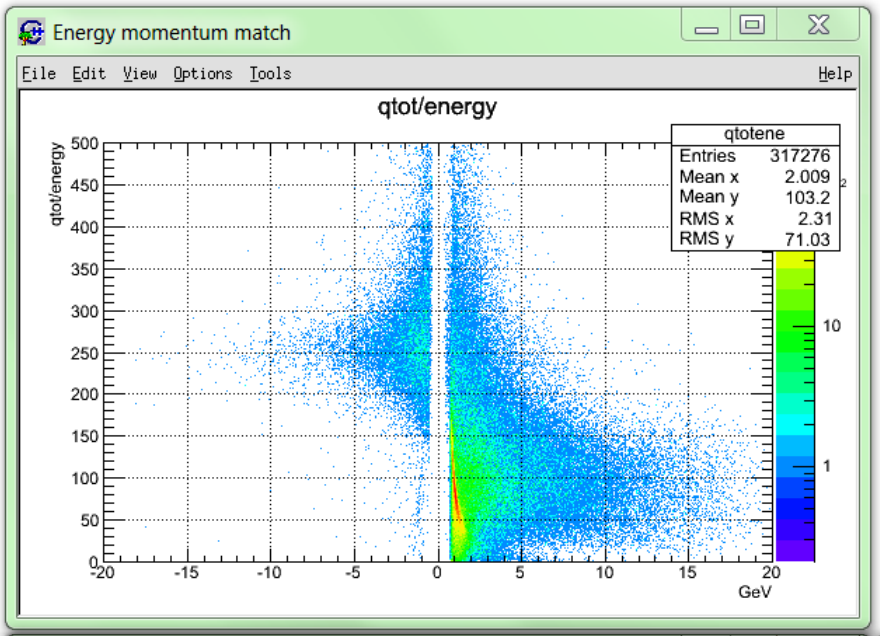
posSelection.C

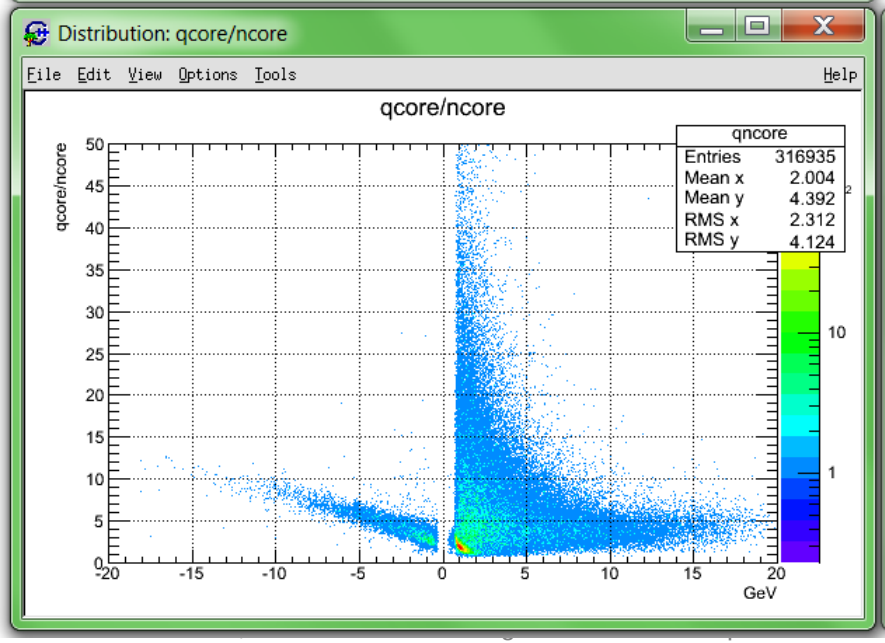
```
#include <TFile.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TF1.h>
#include <TProfile.h>
#include <TGraphAsymmErrors.h>

#include <PamCalo.h>

void posSelection(TString inputFile){

  // Create some histograms.
  // electrons
  TH1D *elesample = new TH1D("elesample","Electron sample;GeV;counts",100,0.,20.);
  TH1D *elesel = new TH1D("elesel","Selected electrons;GeV;counts",100,0.,20.);
  // positrons
  TH1D *possel = new TH1D("possel","Selected positrons;GeV;counts",100,0.,20.);
  // sum ep + em
  TH1D *leptosel = new TH1D("leptpsel","Selected positrons+electrons;GeV;counts",100,0.,20.);

  // scatter plots
  TH2D *qtotene = new TH2D("qtotene","qtot/energy;GeV;qtot/energy",1000,-20.,20.,1000,0.,500.); // energy momentum match
  TH2D *noint = new TH2D("noint","noint;GeV;noint",1000,-20.,20.,1000,0.,35.); // noint distribution
  TH2D *qncore = new TH2D("qncore","qcore/ncore;GeV;qcore/ncore",1000,-20.,20.,5000,0.,50.); // qcore/ncore distribution

  // Functions for selections
  TF1 *fnoint = new TF1("fnoint","abs(0.5+(-100./x))",-30,30);



  // Open the file
  TFile *file = TFile::Open(inputFile);
  TTree *tree = (TTree*)file->Get("pamcalotree");
  PamCalo *pc = new PamCalo();
  tree->SetBranchAddress("PamCalo",&pc);
```

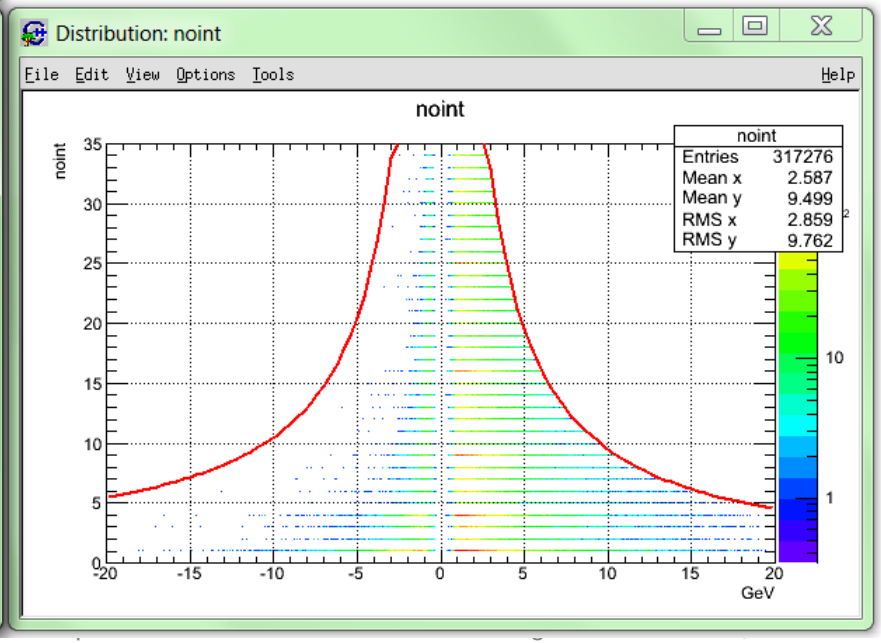
posSelection.C

```
// loop over events
for ( Int_t i=0; i<tree->GetEntries(); i++) {
  //for ( Int_t i=0; i<20000; i++) {

  if ( i%10000 == 0 ) cout << " get entry " << i << "\n";

  tree->GetEntry(i);

  if ( pc->energy < 0. ) elesample->Fill(fabs(pc->energy));

  // selection condition (no selection at the moment)
  if ( pc->noint < fnoint->Eval(pc->energy) // noint selection
       ){

    // fill scatter plots
    noint->Fill(pc->energy, pc->noint);
    if ( pc->ncore > 0 ) qncore->Fill(pc->energy,pc->qcore/(float)pc->ncore);
    if ( pc->energy != 0. ) qtotene->Fill(pc->energy,pc->qtot/fabs(pc->energy));
    // fill histrograms
    if ( pc->energy < 0. ){ // negatives particles
      elesel->Fill(fabs(pc->energy));
    } else { // positive particles
      possel->Fill(fabs(pc->energy));
    }
    leptosel->Fill(fabs(pc->energy));

  }

}

// draw histograms
TCanvas *cqtotenergy = new TCanvas("cqtotenergy","Energy momentum match",200,10,600,400);
cqtotenergy->SetGrid();
cqtotenergy->SetTicks();
cqtotenergy->SetLogz();
cqtotenergy->Draw();
qtotene->Draw("colz");

TCanvas *cnoint = new TCanvas("cnoint","Distribution: noint",200,10,600,400);
cnoint->SetGrid();
cnoint->SetTicks();
cnoint->SetLogz();
cnoint->Draw();
noint->Draw("colz");
fnoint->Draw("same");
```

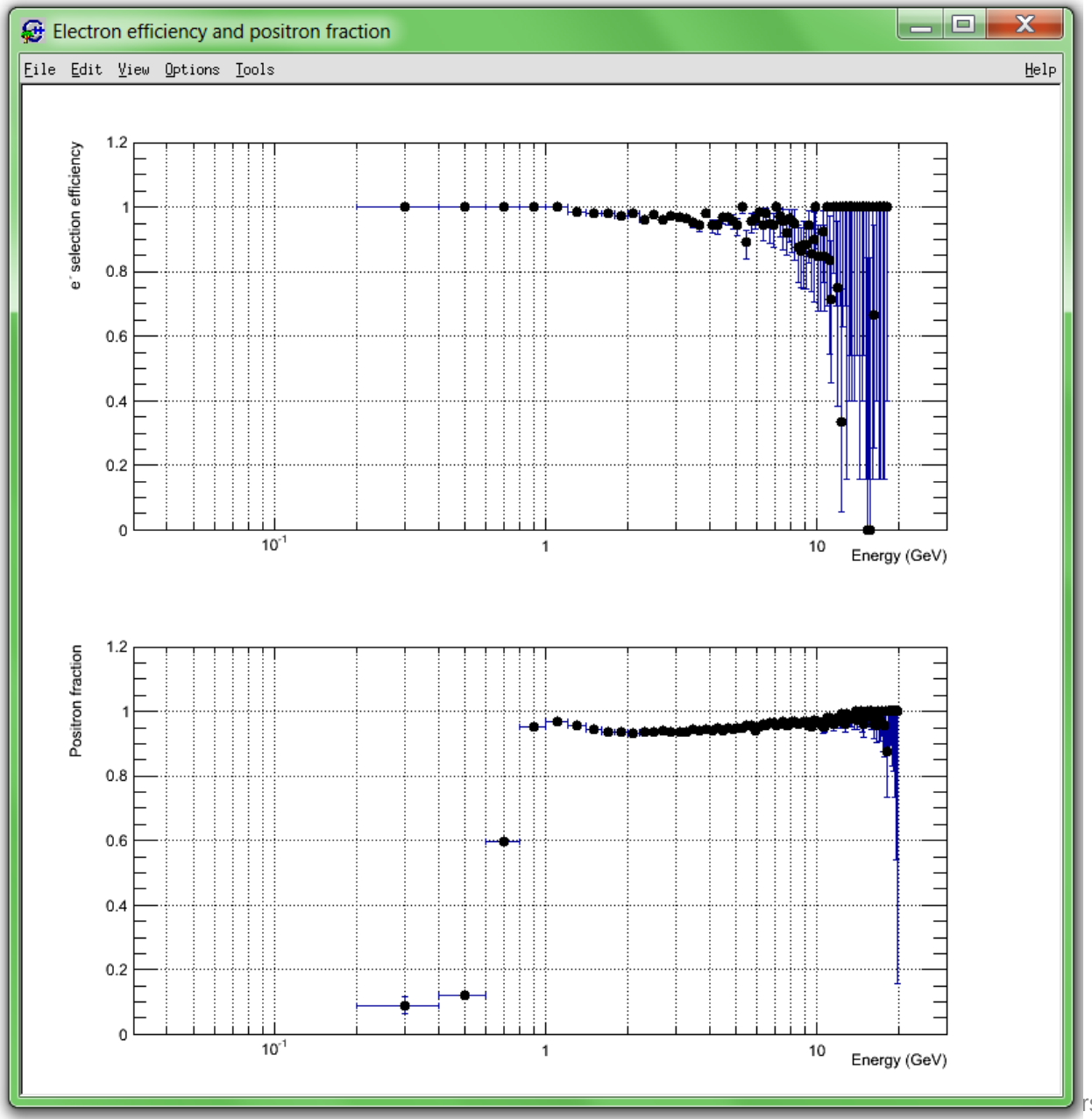
Output plots





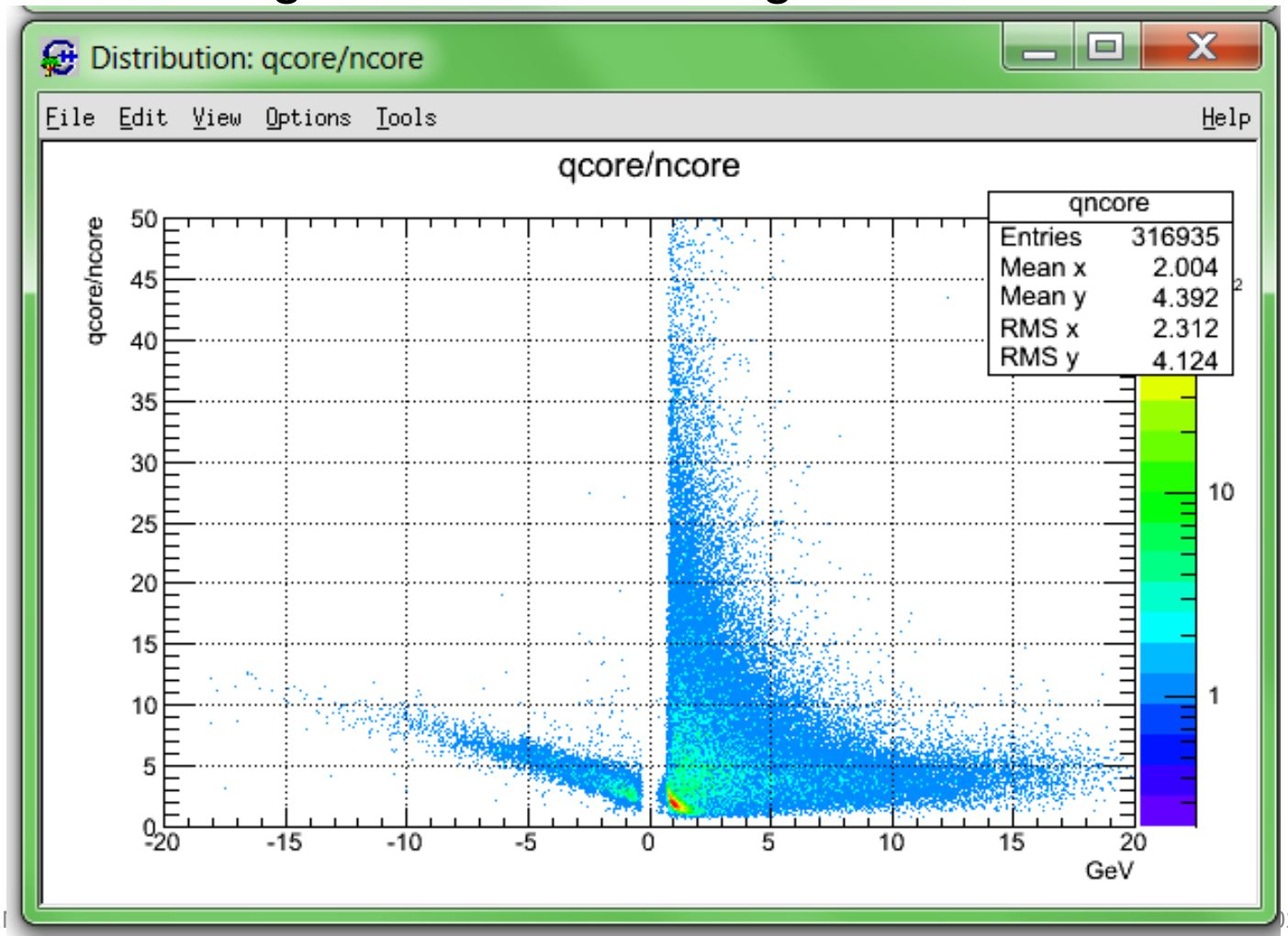


Output plots



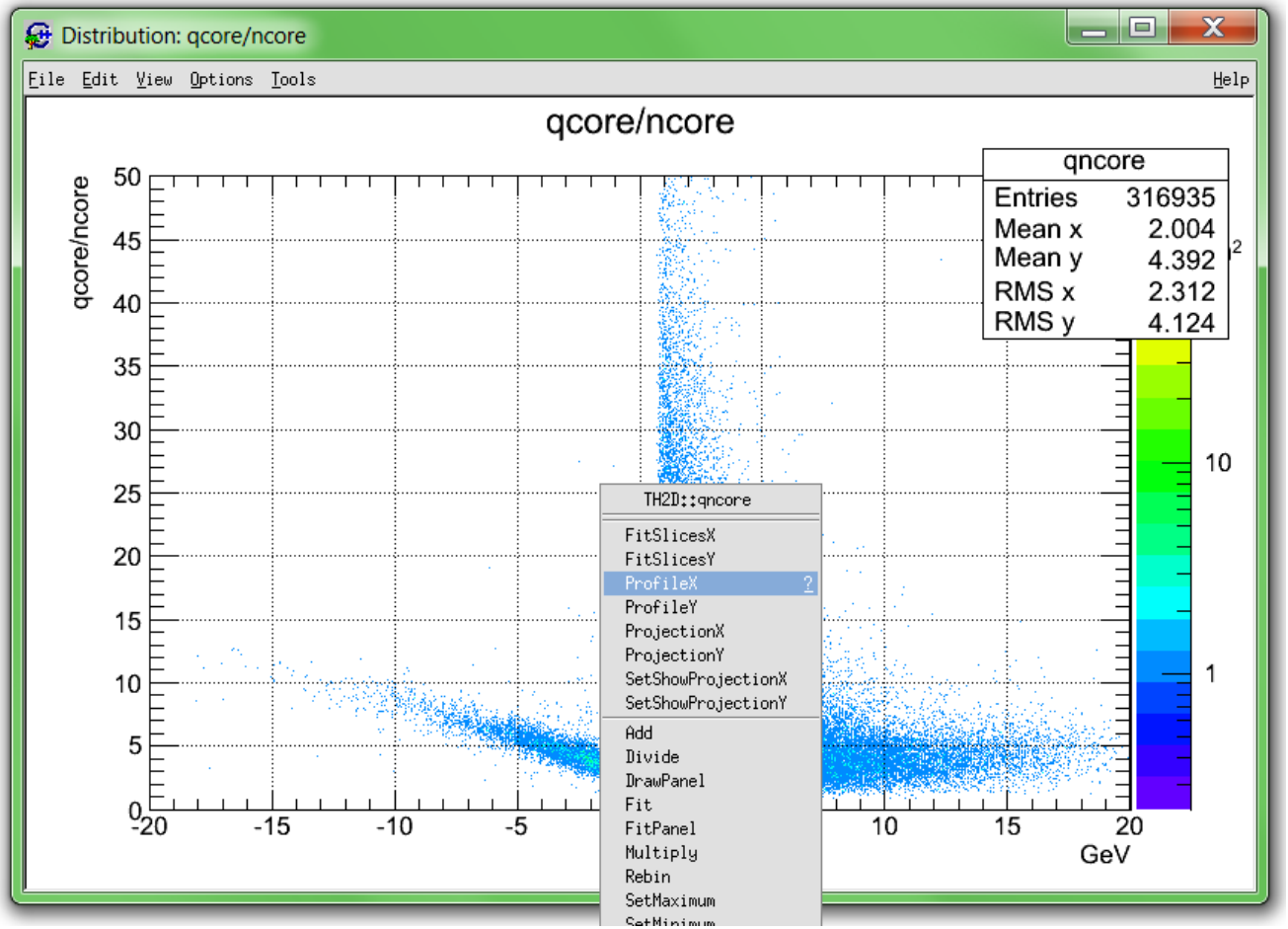
Chosing a cut, case two

Placing a cut using CINT and finding the behaviour of a distribution using TProfile and fitting



Choosing a cut, case two

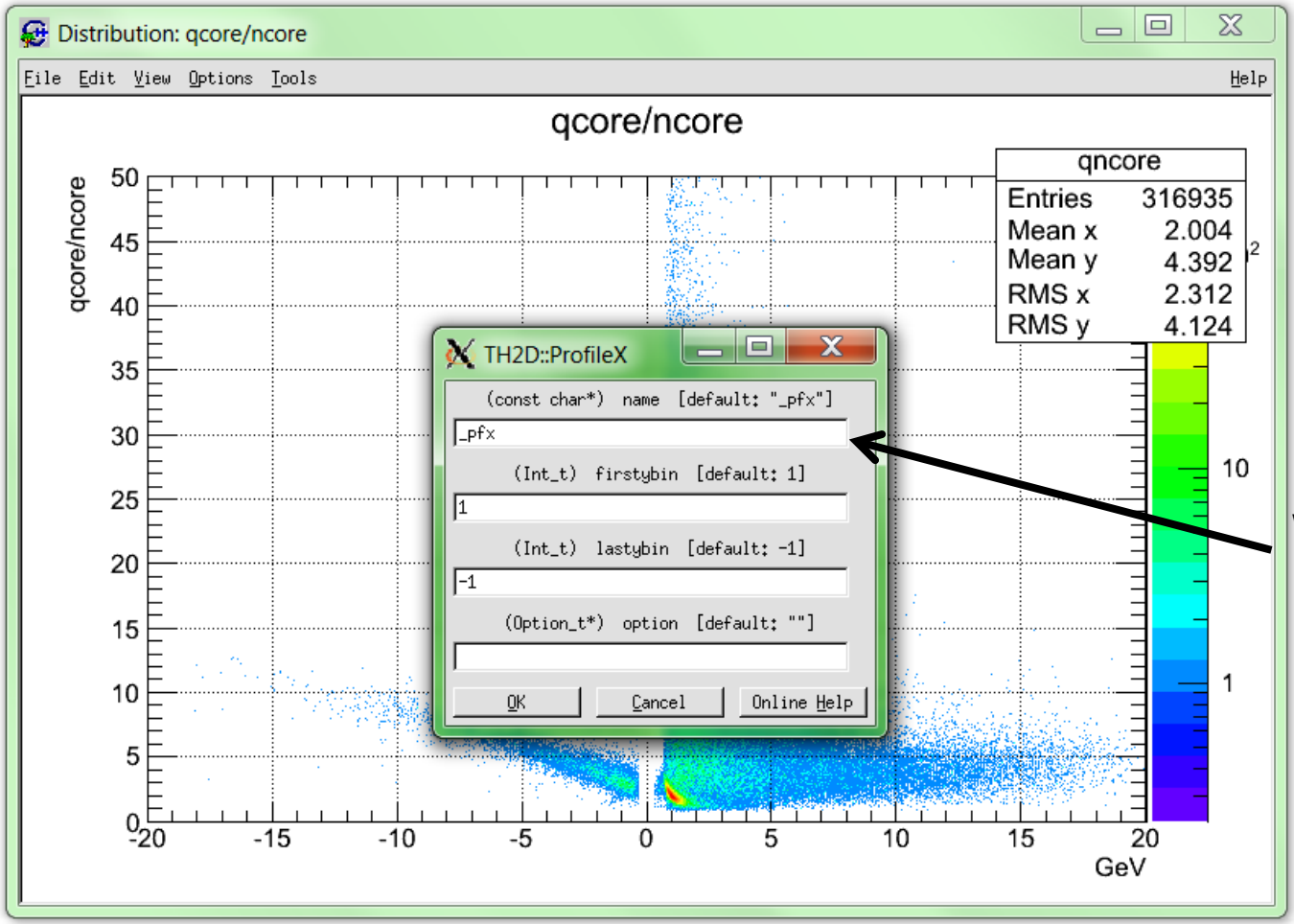
Placing a cut using CINT and finding the behaviour of a distribution using TProfile and fitting



right click on the histogram and choose “ProfileX”

Choosing a cut, case two

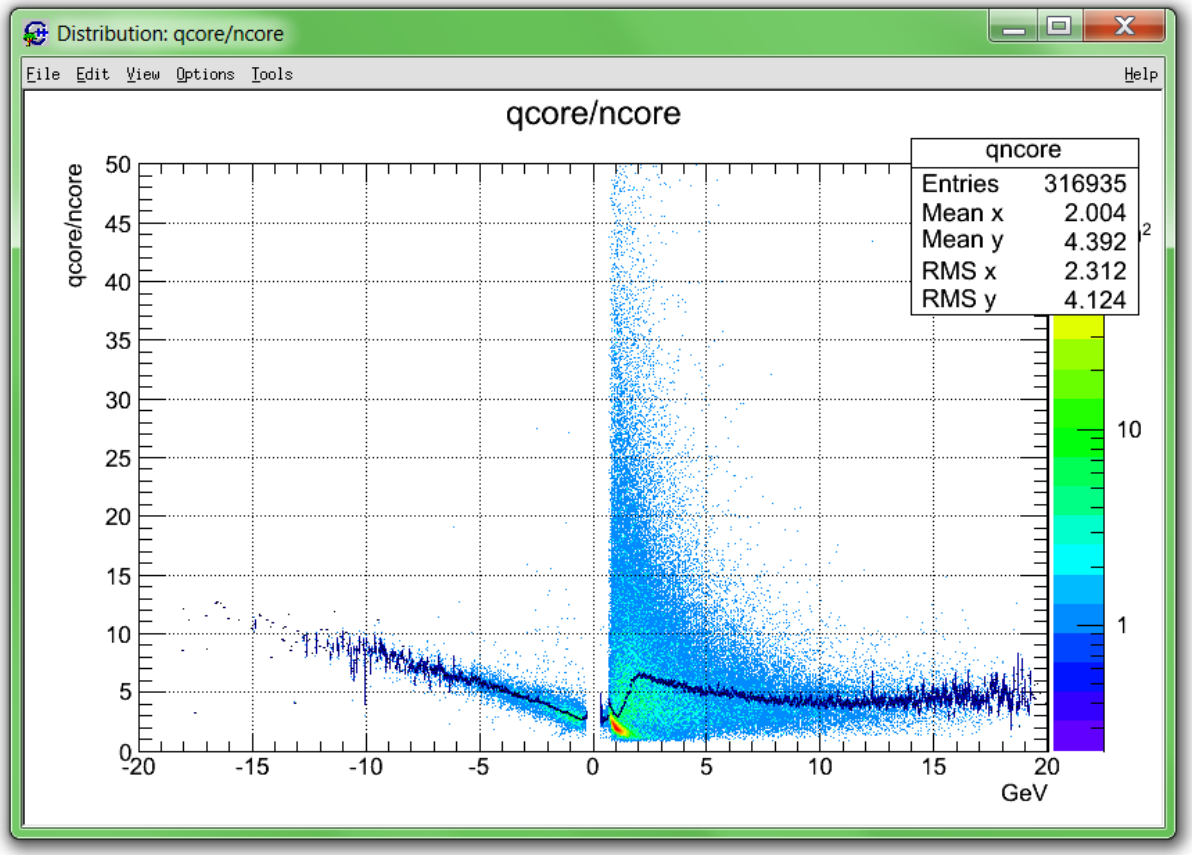
Placing a cut using CINT and finding the behaviour of a distribution using TProfile and fitting



replace "_pfx"
with a name, let's say
"myprof"
click OK

Choosing a cut, case two

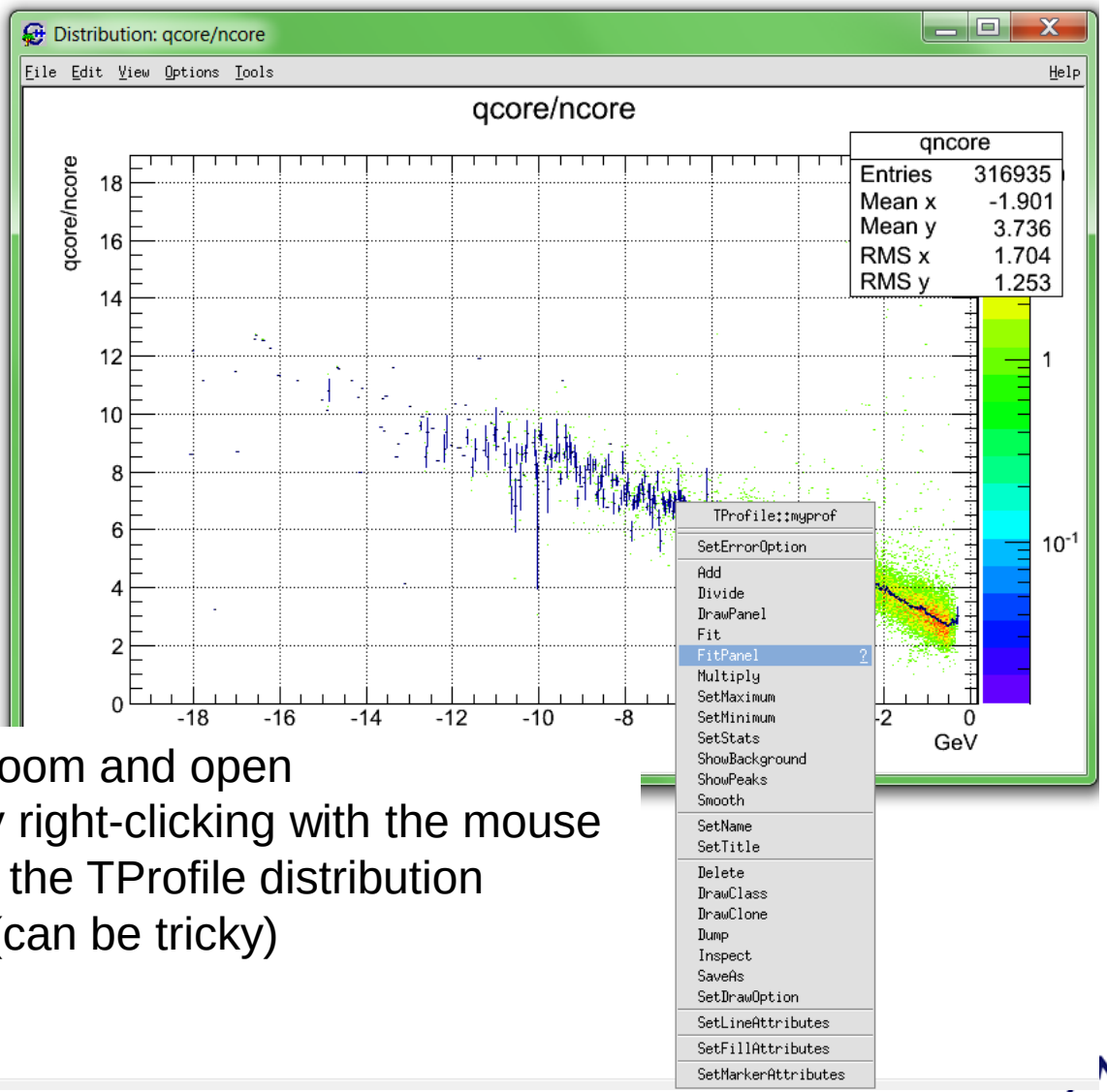
```
get entry 370000
get entry 380000
get entry 390000
get entry 400000
get entry 410000
get entry 420000
get entry 430000
get entry 440000
get entry 450000
get entry 460000
get entry 470000
get entry 480000
get entry 490000
get entry 500000
get entry 510000
get entry 520000
get entry 530000
get entry 540000
get entry 550000
get entry 560000
get entry 570000
get entry 580000
get entry 590000
get entry 600000
get entry 610000
get entry 620000
get entry 630000
get entry 640000
get entry 650000
get entry 660000
get entry 670000
get entry 680000
get entry 690000
get entry 700000
get entry 710000
get entry 720000
get entry 730000
get entry 740000
get entry 750000
get entry 760000
get entry 770000
get entry 780000
get entry 790000
get entry 800000
get entry 810000
get entry 820000
get entry 830000
root [2] myprof->Draw("same")
root [3]
```



draw the profile (even over the scatter plot)

Choosing a cut, case two

```
get entry 380000
get entry 390000
get entry 400000
get entry 410000
get entry 420000
get entry 430000
get entry 440000
get entry 450000
get entry 460000
get entry 470000
get entry 480000
get entry 490000
get entry 500000
get entry 510000
get entry 520000
get entry 530000
get entry 540000
get entry 550000
get entry 560000
get entry 570000
get entry 580000
get entry 590000
get entry 600000
get entry 610000
get entry 620000
get entry 630000
get entry 640000
get entry 650000
get entry 660000
get entry 670000
get entry 680000
get entry 690000
get entry 700000
get entry 710000
get entry 720000
get entry 730000
get entry 740000
get entry 750000
get entry 760000
get entry 770000
get entry 780000
get entry 790000
get entry 800000
get entry 810000
get entry 820000
get entry 830000
root [2] myprof->Draw("same")
root [3]
```



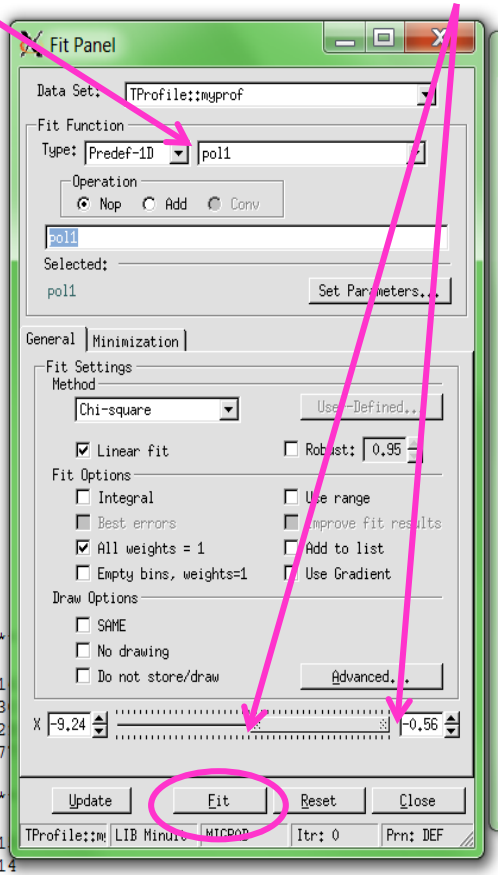
zoom and open
the “FitPanel” by right-clicking with the mouse
when over the TProfile distribution
(can be tricky)

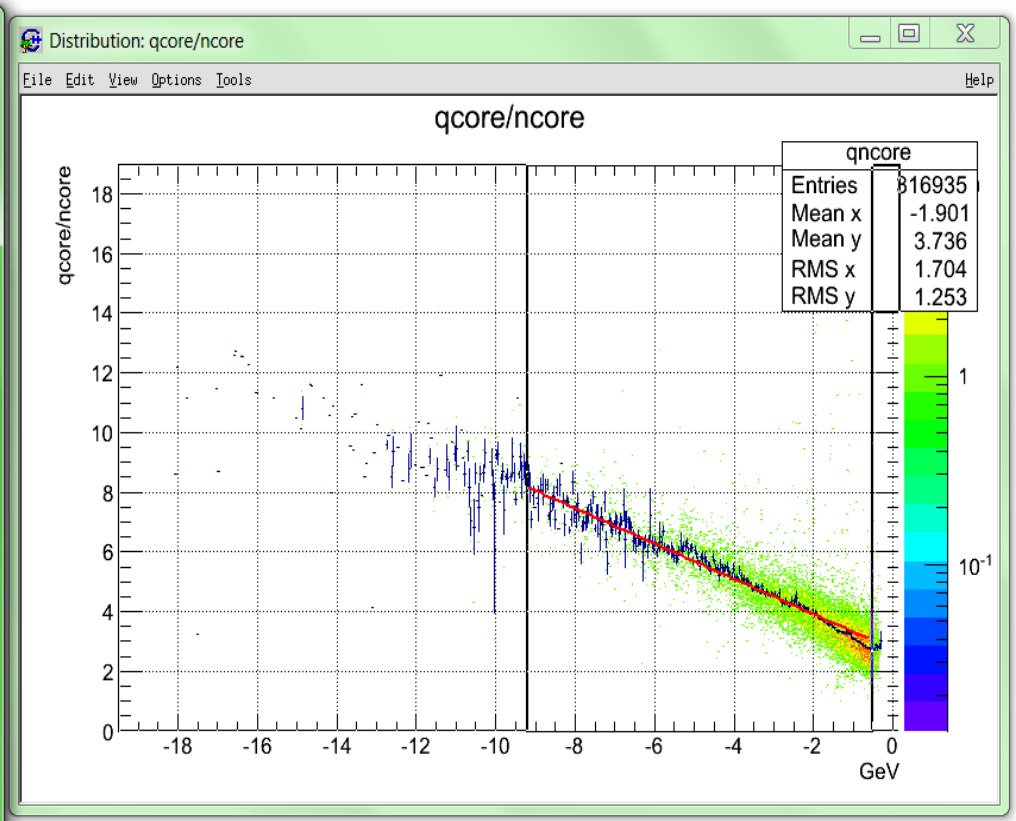
Fitting using the fit panel

choose “pol1”

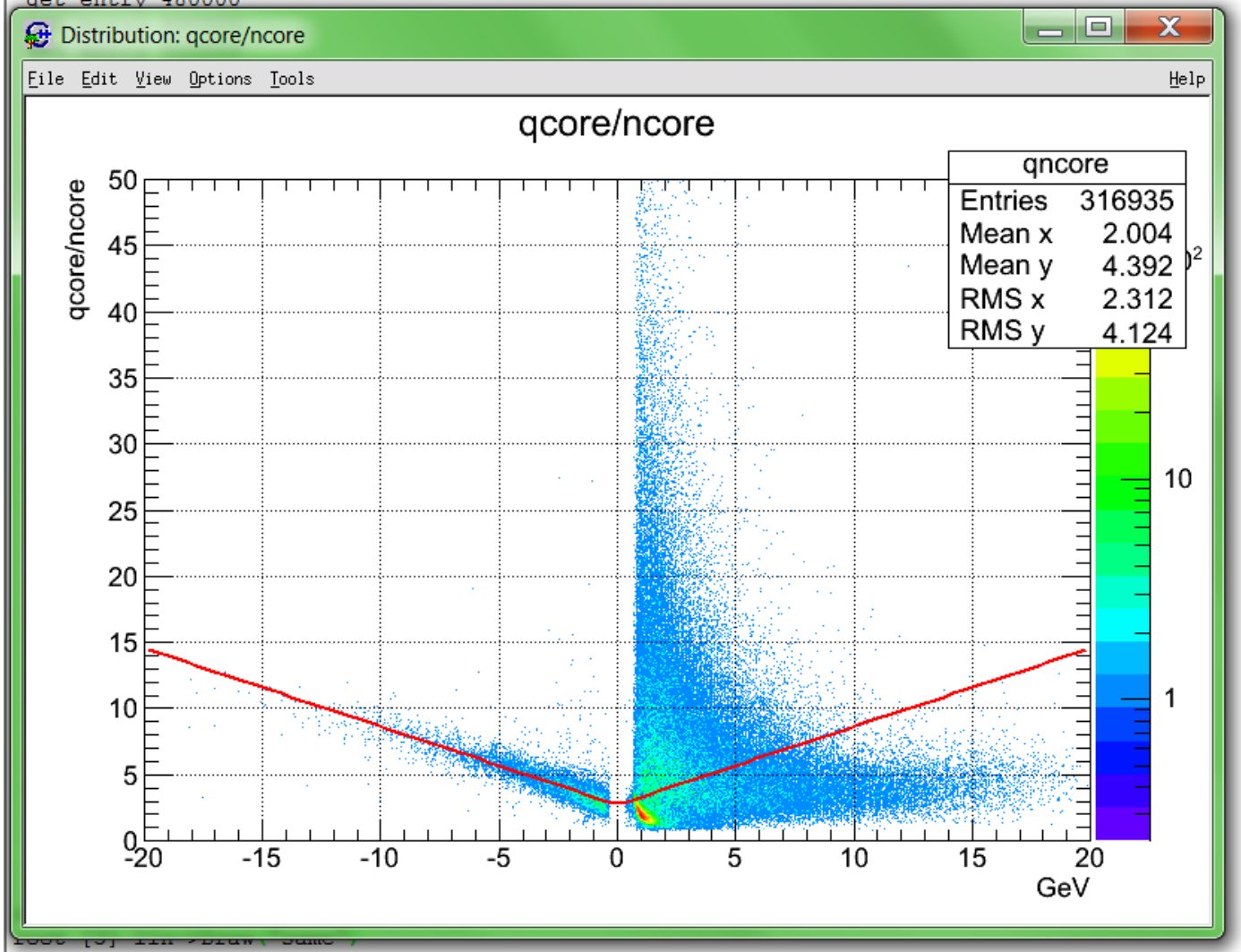
choose the range of fitting (left click and slide)

```
 get entry 580000
 get entry 590000
 get entry 600000
 get entry 610000
 get entry 620000
 get entry 630000
 get entry 640000
 get entry 650000
 get entry 660000
 get entry 670000
 get entry 680000
 get entry 690000
 get entry 700000
 get entry 710000
 get entry 720000
 get entry 730000
 get entry 740000
 get entry 750000
 get entry 760000
 get entry 770000
 get entry 780000
 get entry 790000
 get entry 800000
 get entry 810000
 get entry 820000
 get entry 830000
root [2] myprof->Draw("same")
root [3]
****************************************
Minimizer is Linear
Chi2                      =  1.00221e+1
NDf                       =          23
p0                        =       6.9562
p1                        =     -0.10487

****************************************
Minimizer is Linear
Chi2                      =  7.64498e+1
NDf                       =          214
p0                        =      10.5014   +/-   3.25495e-07
p1                        =      0.31421   +/-   3.80045e-08

****************************************
Minimizer is Linear
Chi2                      =      23.0608
NDf                       =          214
p0                        =       2.7345   +/-   0.149399
p1                        =    -0.591125   +/-   0.0272439
```



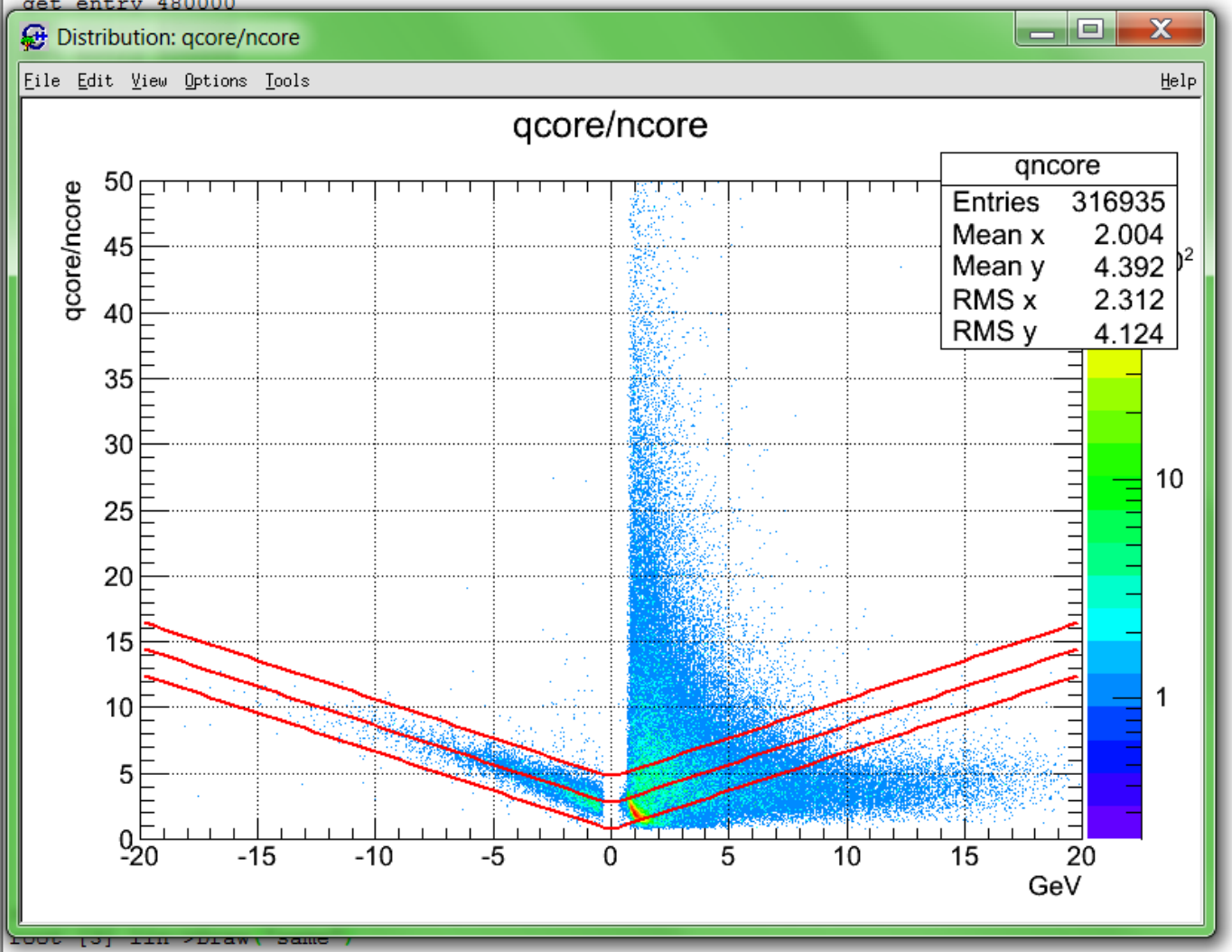


Plotting the function over the whole range



Choosing a region to select

get entry 480000




```
root [4] TF1 *lin = new TF1("lin","2.7345+abs(x)*0.591125",-20,20)
root [5] lin->Draw("same")
root [6] TF1 *lin1 = new TF1("lin1","4.7345+abs(x)*0.591125",-20,20)
root [7] TF1 *lin2 = new TF1("lin2","0.7345+abs(x)*0.591125",-20,20)
root [8] lin1->Draw("same")
root [9] lin2->Draw("same")
root [10]
```

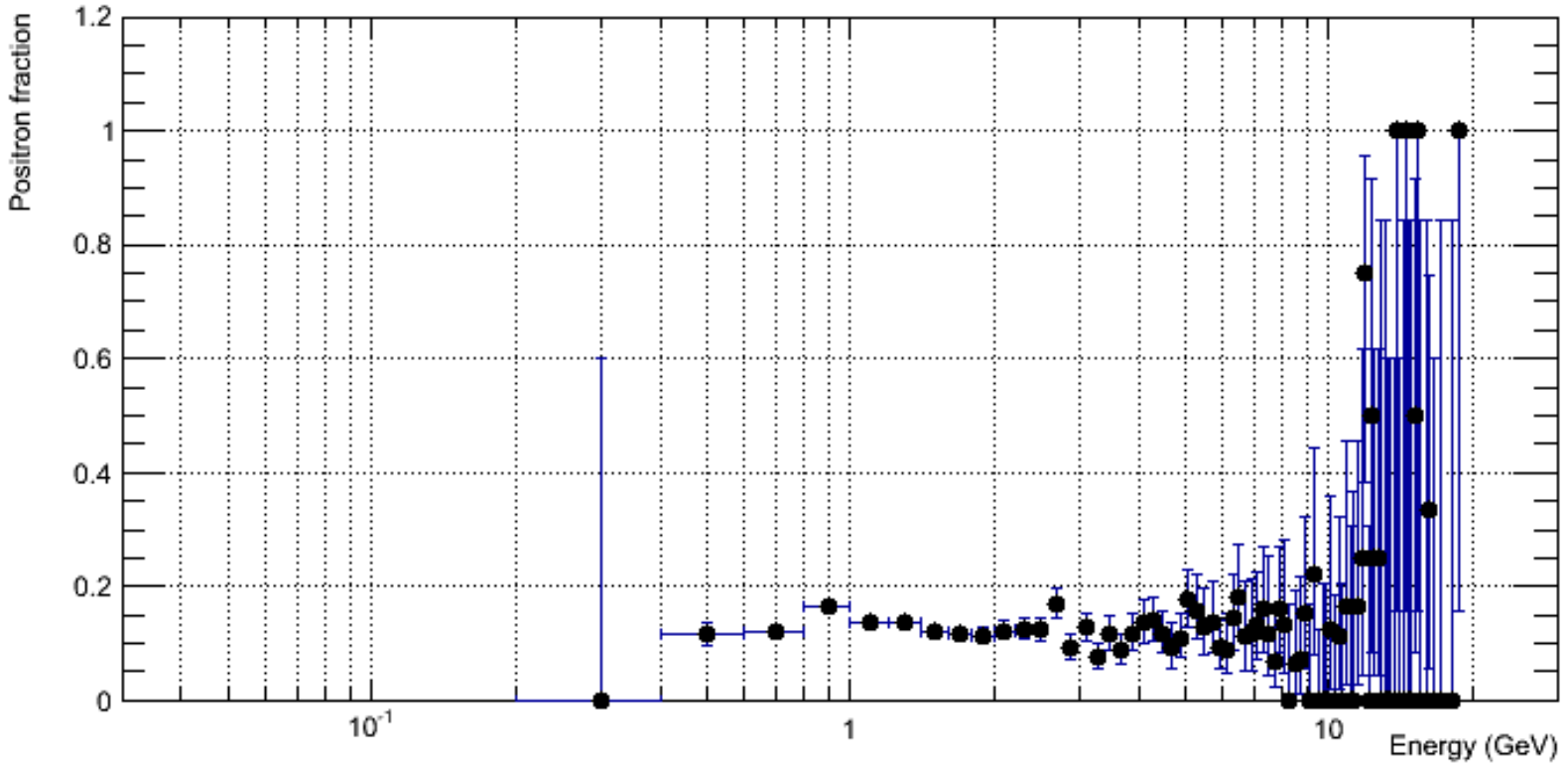
posSelection.C

```
#include <TFile.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TF1.h>
#include <TProfile.h>
#include <TGraphAsymmErrors.h>

#include <PamCalo.h>

void posSelection(TString inputFile){

  // Create some histograms.
  // electrons
  TH1D *elesample = new TH1D("elesample","Electron sample;GeV;counts",100,0.,20.);
  TH1D *elesel = new TH1D("elesel","Selected electrons;GeV;counts",100,0.,20.);
  // positrons
  TH1D *possel = new TH1D("possel","Selected positrons;GeV;counts",100,0.,20.);
  // sum ep + em
  TH1D *leptosel = new TH1D("leptpsel","Selected positrons+electrons;GeV;counts",100,0.,20.);

  // scatter plots
  TH2D *qtotene = new TH2D("qtotene","qtot/energy;GeV;qtot/energy",1000,-20.,20.,1000,0.,500.); // energy momentum match
  TH2D *noint = new TH2D("noint","noint;GeV;noint",1000,-20.,20.,1000,0.,35.); // noint distribution
  TH2D *qncore = new TH2D("qncore","qcore/ncore;GeV;qcore/ncore",1000,-20.,20.,5000,0.,50.); // qcore/ncore distribution

  // Functions for selections
  TF1 *fnoint = new TF1("fnoint","abs(0.5+(-100./x))",-30,30);

  TF1 *fqncorelow = new TF1("fqncorelow","0.111+0.44751*abs(x)",-30,30);
  TF1 *fqncoreup = new TF1("fqncoreup","8.111+0.44751*abs(x)",-30,30);

  // Open the file
  TFile *file = TFile::Open(inputFile);
  TTree *tree = (TTree*)file->Get("pamcalotree");
  PamCalo *pc = new PamCalo();
  tree->SetBranchAddress("PamCalo",&pc);
```

posSelection.C

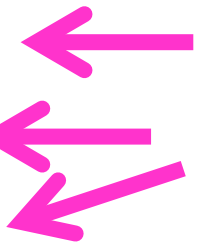
```
// loop over events
for ( Int_t i=0; i<tree->GetEntries(); i++) {
  //for ( Int_t i=0; i<20000; i++) {

  if ( i%10000 == 0 ) cout << " get entry " << i << "\n";

  tree->GetEntry(i);

  if ( pc->energy < 0. ) elesample->Fill(fabs(pc->energy));

  // selection condition (no selection at the moment)
  if ( pc->noint < fnoint->Eval(pc->energy) // noint selection
    && pc->ncore > 0 && pc->qcore/pc->ncore > fqncorelow->Eval(pc->energy) &&  pc->qcore/pc->ncore < fqncoreup->Eval(pc->energy) // qncore selection
       ){

    // fill scatter plots
    noint->Fill(pc->energy, pc->noint);
    if ( pc->ncore > 0 ) qncore->Fill(pc->energy,pc->qcore/(float)pc->ncore);
    if ( pc->energy != 0. ) qtotene->Fill(pc->energy,pc->qtot/fabs(pc->energy));
    // fill histrograms
    if ( pc->energy < 0. ){ // negatives particles
      elesel->Fill(fabs(pc->energy));
    } else { // positive particles
      possel->Fill(fabs(pc->energy));
    }
    leptosel->Fill(fabs(pc->energy));

  }

}


TCanvas *cqncore = new TCanvas("cqncore","Distribution: qcore/ncore",200,10,600,400);
cqncore->SetGrid();
cqncore->SetTicks();
cqncore->SetLogz();
cqncore->Draw();
qncore->Draw("colz");
fqncorelow->Draw("same");
fqncoreup->Draw("same");
```

Output plots



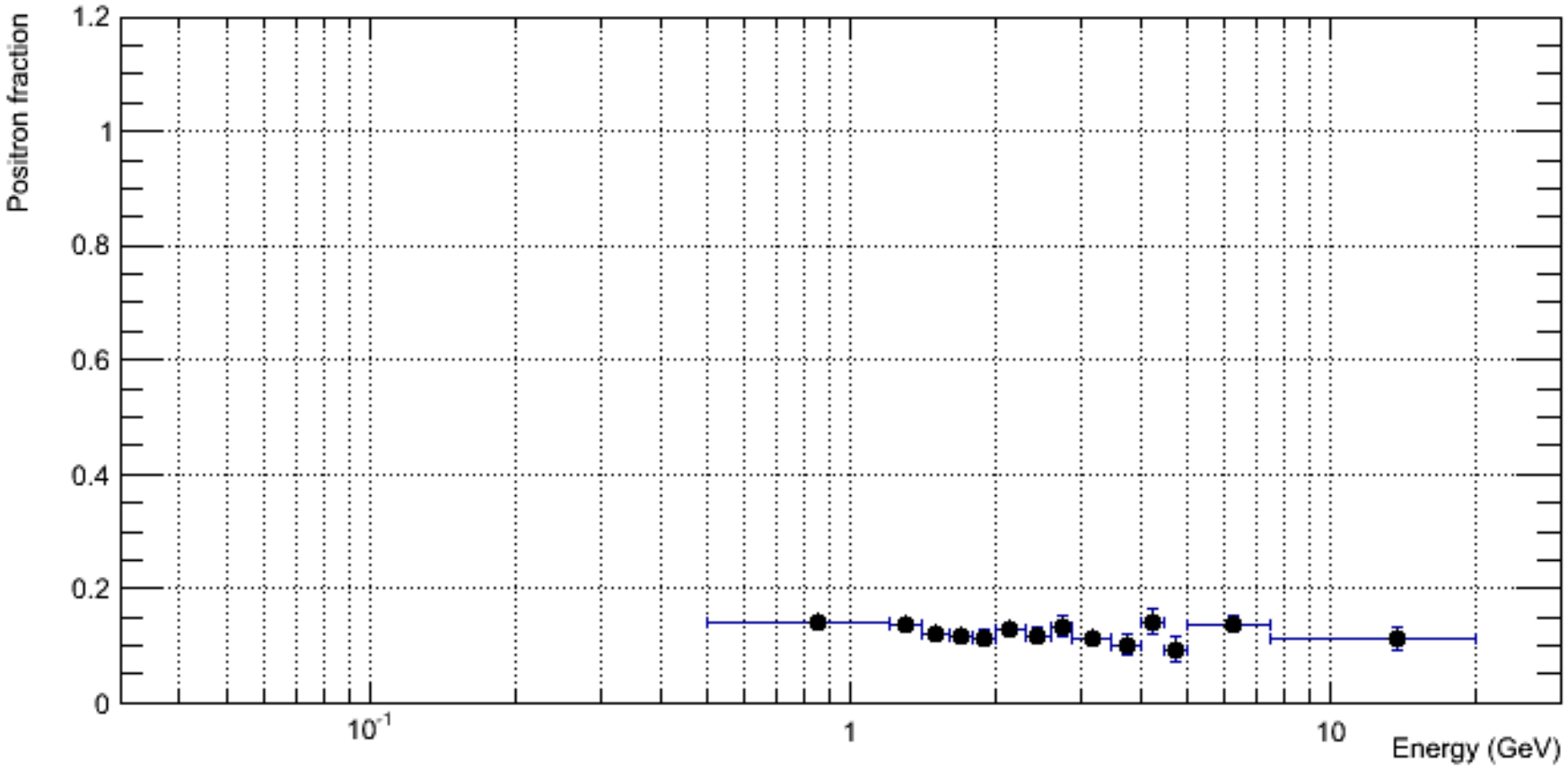
posSelection.C: TH arbitrary binning

```
void posSelection(TString inputFile){

  Float_t xbin[15]={0.5,1.2,1.4,1.6,1.8,2.,2.3,2.6,2.9,3.5,4.,4.5,5.,7.5,20.};
  // Create some histograms.
  // electrons
  TH1D *elesample = new TH1D("elesample","Electron sample;GeV;counts",14,xbin);
  TH1D *elesel = new TH1D("elesel","Selected electrons;GeV;counts",14,xbin);
  // positrons
  TH1D *possel = new TH1D("possel","Selected positrons;GeV;counts",14,xbin);
  // sum ep + em
  TH1D *leptosel = new TH1D("leptpsel","Selected positrons+electrons;GeV;counts",14,xbin);

  // scatter plots
  TH2D *qtotene = new TH2D("qtotene","qtot/energy;GeV;qtot/energy",1000,-20.,20.,1000,0.,500.); // energy momentum match
  TH2D *noint = new TH2D("noint","noint;GeV;noint",1000,-20.,20.,1000,0.,35.); // noint distribution
  TH2D *qncore = new TH2D("qncore","qcore/ncore;GeV;qcore/ncore",1000,-20.,20.,5000,0.,50.); // qcore/ncore distribution

  // Functions for selections
  TF1 *fnoint = new TF1("fnoint","abs(0.5+(-100./x))",-30,30);
```



http://root.cern.ch/root/html532/TH1D.html#TH1D:TH1D@2

Output plots



TTree:MakeClass()

```
|Emi@marte ~>root /home/mocchiut/pamela/data/pamelasimu.root
root [0]
Attaching file /home/mocchiut/pamela/data/pamelasimu.root as _file0...
root [1] pamcalotree->MakeClass()
Info in <TTreePlayer::MakeClass>: Files: pamcalotree.h and
pamcalotree.C generated from TTree: pamcalotree
(Int_t)0
root [2]
```

TTree:MakeClass()

pamcalotree.C

```
#define pamcalotree_cxx
#include "pamcalotree.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

void pamcalotree::Loop()
{
//   In a ROOT session, you can do:
//      Root > .L pamcalotree.C
//      Root > pamcalotree t
//      Root > t.GetEntry(12); // Fill t data members with entry number 12
//      Root > t.Show();       // Show values of entry 12
//      Root > t.Show(16);     // Read and show values of entry 16
//      Root > t.Loop();       // Loop on all entries
//

//     This is the loop skeleton where:
//    jentry is the global entry number in the chain
//    ientry is the entry number in the current Tree
//  Note that the argument to GetEntry must be:
//    jentry for TChain::GetEntry
//    ientry for TTree::GetEntry and TBranch::GetEntry
//
//       To read only selected branches, Insert statements like:
// METHOD1:
//    fChain->SetBranchStatus("*",0);  // disable all branches
//    fChain->SetBranchStatus("branchname",1);  // activate branchname
// METHOD2: replace line
//    fChain->GetEntry(jentry);       //read all branches
//by  b_branchname->GetEntry(ientry); //read only this branch
   if (fChain == 0) return;

   Long64_t nentries = fChain->GetEntriesFast();

   Long64_t nbytes = 0, nb = 0;
   for (Long64_t jentry=0; jentry<nentries;jentry++) {
      Long64_t ientry = LoadTree(jentry);
      if (ientry < 0) break;
      nb = fChain->GetEntry(jentry);   nbytes += nb;
      // if (Cut(ientry) < 0) continue;
   }
}
```

TTree:MakeClass()

pamcalotree.C | *pamcalotree.h

```
//////////////////////////////////////////////////////////
// This class has been automatically generated on
// Thu May 28 16:37:56 2015 by ROOT version 5.32/00
// from TTree pamcalotree/PAMELA simulations
// found on file: /home/mocchiut/pamela/data/pamsimu2014.root
//////////////////////////////////////////////////////////
#ifndef pamcalotree_h
#define pamcalotree_h
#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>
// Header file for the classes stored in the TTree if any.
#include "./PamCalo.h"
#include <TObject.h>

// Fixed size dimensions of array or collections stored in the TTree if any.
class pamcalotree {
public :
   TTree          *fChain;   //!pointer to the analyzed TTree or TChain
   Int_t           fCurrent; //!current Tree number in a TChain

   // Declaration of leaf types
 //PamCalo         *PamCalo;
   UInt_t          fUniqueID;
   UInt_t          fBits;
   UInt_t          time;
   Float_t         energy;
   Float_t         beta;
   Float_t         dEdx;
   Int_t           pID;
   Float_t         qtot;
   [...]

   // List of branches
   TBranch        *b_PamCalo_fUniqueID;   //!
   TBranch        *b_PamCalo_fBits;   //!
   TBranch        *b_PamCalo_time;   //!
   TBranch        *b_PamCalo_energy;   //!
   TBranch        *b_PamCalo_beta;   //!
   [...]

   pamcalotree(TTree *tree=0);
   virtual ~pamcalotree();
```

TTree:MakeClass()

pamcalotree.C | *pamcalotree.h

```
   virtual ~pamcalotree();
   virtual Int_t    Cut(Long64_t entry);
   virtual Int_t    GetEntry(Long64_t entry);
   virtual Long64_t LoadTree(Long64_t entry);
   virtual void     Init(TTree *tree);
   virtual void     Loop();
   virtual Bool_t   Notify();
   virtual void     Show(Long64_t entry = -1);
};

#endif

#ifdef pamcalotree_cxx
pamcalotree::pamcalotree(TTree *tree) : fChain(0)
{
// if parameter tree is not specified (or zero), connect the file
// used to generate this class and read the Tree.
   if (tree == 0) {
      TFile *f = (TFile*)gROOT->GetListOfFiles()->FindObject("/home/mocchiut/pamela/data/
pamsimu2014.root");
      if (!f || !f->IsOpen()) {
         f = new TFile("/home/mocchiut/pamela/data/pamsimu2014.root");
      }
      f->GetObject("pamcalotree",tree);

   }
   Init(tree);
}

pamcalotree::~pamcalotree()
{
   if (!fChain) return;
   delete fChain->GetCurrentFile();
}

Int_t pamcalotree::GetEntry(Long64_t entry)
{
// Read contents of entry.
   if (!fChain) return 0;
   return fChain->GetEntry(entry);
}
Long64_t pamcalotree::LoadTree(Long64_t entry)
```

TTree:MakeClass()

pamcalotree.C | *pamcalotree.h

```
Long64_t pamcalotree::LoadTree(Long64_t entry)
{
// Set the environment to read one entry
   if (!fChain) return -5;
   Long64_t centry = fChain->LoadTree(entry);
   if (centry < 0) return centry;
   if (fChain->GetTreeNumber() != fCurrent) {
      fCurrent = fChain->GetTreeNumber();
      Notify();
   }
   return centry;
}

void pamcalotree::Init(TTree *tree)
{
   // The Init() function is called when the selector needs to initialize
   // a new tree or chain. Typically here the branch addresses and branch
   // pointers of the tree will be set.
   // It is normally not necessary to make changes to the generated
   // code, but the routine can be extended by the user if needed.
   // Init() will be called many times when running on PROOF
   // (once per file to be processed).

   // Set branch addresses and branch pointers
   if (!tree) return;
   fChain = tree;
   fCurrent = -1;
   fChain->SetMakeClass(1);

   fChain->SetBranchAddress("fUniqueID", &fUniqueID, &b_PamCalo_fUniqueID);
   fChain->SetBranchAddress("fBits", &fBits, &b_PamCalo_fBits);
   fChain->SetBranchAddress("time", &time, &b_PamCalo_time);
   fChain->SetBranchAddress("energy", &energy, &b_PamCalo_energy);
   fChain->SetBranchAddress("beta", &beta, &b_PamCalo_beta);
   [...]

   Notify();
}

Bool_t pamcalotree::Notify()
{
   // The Notify() function is called when a new file is opened. This
   // can be either for a new TTree in a TChain or when when a new TTree
```

TTree:MakeClass()

```
Bool_t pamcalotree::Notify()
{
   // The Notify() function is called when a new file is opened. This
   // can be either for a new TTree in a TChain or when when a new TTree
   // is started when using PROOF. It is normally not necessary to make changes
   // to the generated code, but the routine can be extended by the
   // user if needed. The return value is currently not used.

   return kTRUE;
}

void pamcalotree::Show(Long64_t entry)
{
// Print contents of entry.
// If entry is not specified, print current entry
   if (!fChain) return;
   fChain->Show(entry);
}
Int_t pamcalotree::Cut(Long64_t entry)
{
// This function may be called from Loop.
// returns  1 if entry is accepted.
// returns -1 otherwise.
   return 1;
}
#endif // #ifdef pamcalotree_cxx
```

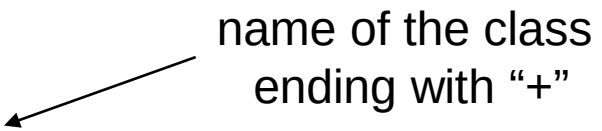
How to compile a ROOTable lib

Step 0. Create a class using ClassDef and ClassImp directives (look at PamCalo for examples)

How to compile a ROOTable lib

Step 1. Create a new file which name ends in "LinkDef.h" (e.g. PamCaloLinkDef.h) containing something like:

```
#ifdef __CINT__
#pragma link off all class;
#pragma link off all function;
#pragma link off all global;
#pragma link off all typdef;
#pragma link C++ class PamCalo+;
#endif
```

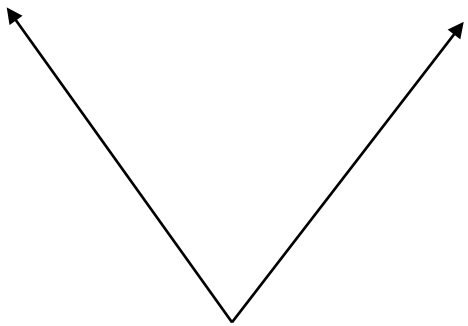


https://gcc.gnu.org/onlinedocs/cpp/Pragmas.html

How to compile a ROOTable lib

Step 2. Create a new cpp file (so called "dictionary file" e.g. PamCaloDict.cpp) using rootcint:

```
bash> rootcint –f PamCaloDict.cpp –c –I`root-
 config --incdir` PamCalo.h PamCaloLinkDef.h
```



both header of the class AND LinkDef.h files

How to compile a ROOTable lib

Step 3. Create the objects for dictionary file and class implementation file as usual:

```
bash> g++ -Wall -fPIC –I./ –I`root-config --incdir` -c PamCalo.cpp -o PamCalo.o
```

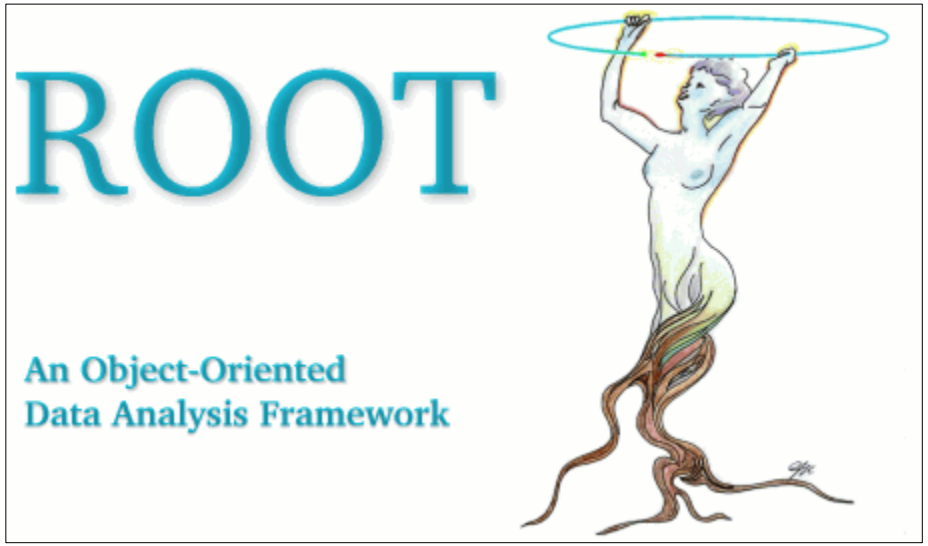
```
bash> g++ -Wall -fPIC –I./ –I`root-config --incdir` -c PamCaloDict.cpp -o PamCaloDict.o
```

How to compile a ROOTable lib

Step 4. Create the shared lib merging both the class implementation and the dictionary objects:

```
g++ -Wall -shared -o libPamCalo.so PamCalo.o
 PamCaloDict.o
```

ROOT 5 vs ROOT 6



ROOT 5 vs ROOT 6

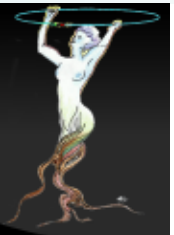
Do we need yet another custom C++ interpreter? (Axel Neumann)
"A ROOT User" asks "Is it really necessary to replace CINT dictionary with cling?", bringing up very reasonable concerns and arguments against re-implementing CINT. I will try to answer his comments to clarify why we do it, and how it connects with the rest.
A fundamental misconception is that the status quo is acceptable. It is not, for several reasons.

CINT vs C++
CINT was designed (25 years ago!) to be a C interpreter; C++ support was added later. It still has many shortcomings with C++ 2003, let alone C++11.

CINT maintenance
The original author of CINT, Masaharu Goto, has moved on; CINT has been maintained mainly by the ROOT team. It has 300k lines of code; that's a considerable fraction of ROOT's. It has been designed to fit into an integrated processing unit of appliances (like medical ones) - not for 16GB RAM, 8 compute thread, 50000 class environments.

ROOT 5 vs ROOT 6

Cling Is Better Than CINT



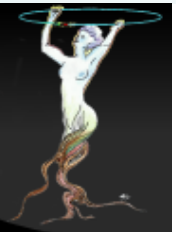
- Full C++ support
  - STL + templates
  - Path to C++11
- Correctness
- Better type information and representations
- Always compile in memory
- Much less code to maintain

```
****************** CLING ******************
* Type C++ code and press enter to run it *
*             Type .q to exit             *
*******************************************
[cling]$ #include <vector>
[cling]$ #include <map>
[cling]$ #include <string>
[cling]$ #include <set>
[cling]$ using namespace std;
[cling]$ vector<map<string,set<int> > > a
(vector<map<string, set<int> > >) @0x10b190020
[cling]$
```

vvassilev / CHEP 2012.05.21 4

ROOT 5 vs ROOT 6

Cling's Dual Personality



- An interpreter – looks like an interpreter and behaves like an interpreter
  Cling follows the read-evaluate-print-loop (repl) concept.
- More than interpreter – built on top of compiler libraries (Clang and LLVM)
  Contains interpreter parts and compiler parts. More of an interactive compiler or an interactive compiler interface for clang.

No need to compile Cling/ROOT with Clang/LLVM

vvassilev / CHEP 2012.05.21 5

ROOT 5 vs ROOT 6

ROOT 5 vs ROOT 6

Challenges

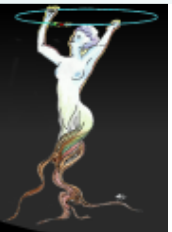
- *Error recovery*
  Even though the user has typed wrong input at the prompt cling must survive, i.e issue an error and continue to work.
- Initialization of global variables
  Cling depends on global variables, which need to be initialized. However, the global variables continue to be added (potentially) with every input line.
- Late binding
  Cling needs to provide a way for symbols unavailable at compile-time a second chance to be provided at runtime.
- Value printer
  The interactive mode obeys the repl concept and there should be way of easy print value and type of expression in a user-extensible way.
- Running statements
  CINT-specific C++ extension improving the user interaction with the interpreter from the terminal.

IMPLEMENTED

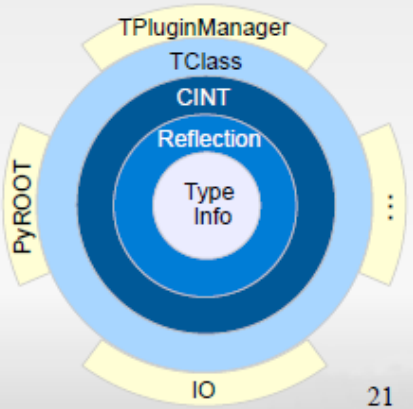
vvassilev / CHEP 2012.05.21 18

ROOT 5 vs ROOT 6

Dictionaries



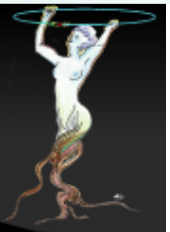
- Describe compiled code
  Incarnation of the type information and reflection in ROOT. The only way of crossing the border between interpretation and compilation.
- CINT and Reflex dictionaries:
  - Double the size of the libraries
  - Multiple copies of the dictionary data in the memory
  - Incompatible reflection formats
  - Do not cover 100% of C++



vvassilev / CHEP 2012.05.21 21

ROOT 5 vs ROOT 6

CINT Dictionary Use

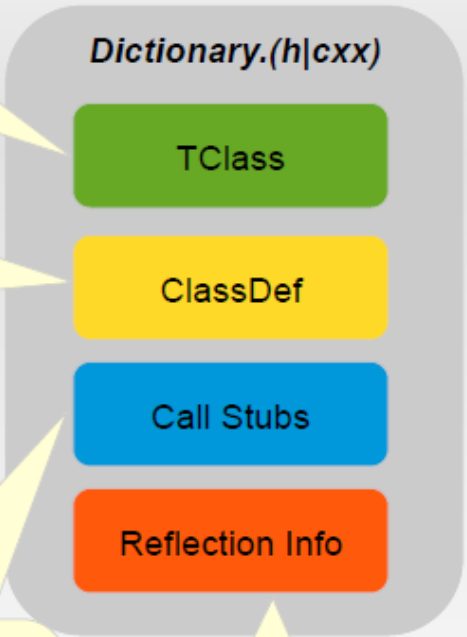


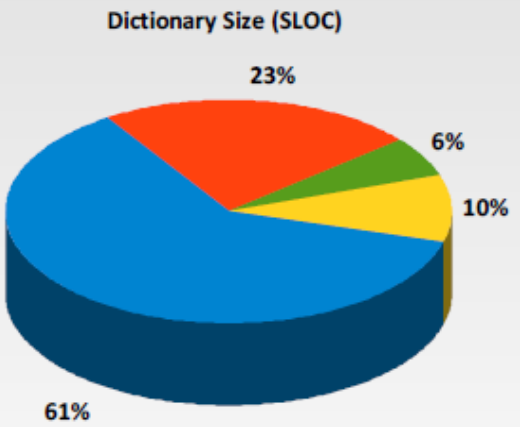
Source of ROOT's class description

Member functions injected by the ClassDef macro

Target-independent, normalized signatures used to call compiled functions.

Teaches CINT what a class contains



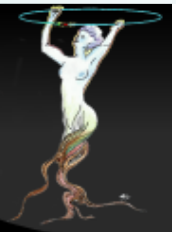


vvassilev / CHEP 2012.05.21

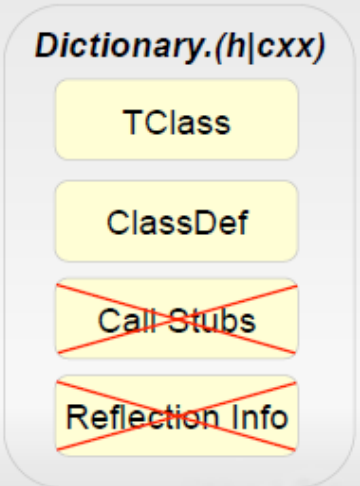
22

ROOT 5 vs ROOT 6

ROOT 6.x



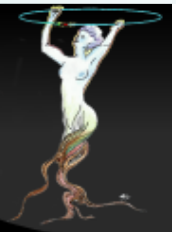
- Reduce size of dictionaries
  - Mid term: Call Stubs & Reflection Info goes away!
  - Long term: No dictionaries at all
- Compiled TFormula



vvassilev / CHEP 2012.05.21 25

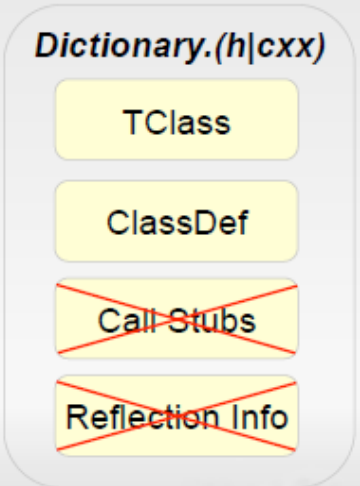
ROOT 5 vs ROOT 6

ROOT 6.x



- Reduce size of dictionaries
  - Mid term: Call Stubs & Reflection Info goes away!
  - Long term: No dictionaries at all
- Compiled TFormula

“rootcint” replaced by “rootcling” on the command line!



vvassilev / CHEP 2012.05.21 25

ROOT 5 vs ROOT 6

Backward Compatibility with CINT

We will discontinue support for using '.' and '->' interchangeably. Note that references have the identical "performance issue" as pointer derefs and use '.' and that '.' often results in a memory load, too, so I don't take performance differences as an argument :)

But we want to encourage proper C++. We do provide most of the interpreter extensions, but they can be turned off.

Class Hierarchy

A pet peeve of many, where it's not even clear how much your average novice physicist really cares. None of the real in contrast to "I'd rather want to do computing" physicists like me :) Physicists I talked to ever saw that as an issue.

Axel Naumann

ROOT 5 vs ROOT 6

Templates

ROOT was limited in the use of templates due to the way CINT called functions. Now that we have a just in time compiler we will be able to instantiate templates at runtime, and call their functions as needed. That's e.g. the main reason why the TTreeReader (https://sft.its.cern.ch/jira/browse/ROOT5165) only really works well in ROOT6. We will migrate interfaces where it's useful, but due to the lack of documentability of possible template arguments (where concepts would help), novices usually find templates repelling compared to traditional classes.

C++11 (NB: not supported by gcc < 4.8; default is C98 for gcc < 6.1)

There are several parts to it: ROOT needs to be able to parse C++11 code, and ROOT 6 will be able to do that out of the box when built with C++11 turned in. The next part is the accessibility of C++11 features through its reflection interfaces (e.g. TClass) that will be done after ROOT 6. And the last part of C++11 support is adding features to ROOT's I/O where needed.

Axel Naumann

Exercises

The following file (52 MB)

`pamelasimu.root`

contains the TTree pamcalotree, storing data with the PamCalo class.

Variables from PAMELA simulation.

About 800.000 events: protons, anti-protons, electrons and positrons mixed together. In this file:

Antiprotons to protons ratio: about 0.0006

Positrons to electrons ratio: about 0.12

Electrons to protons: is about 0.003

Exercises

```
bash> root
root [0] gSystem->Load("/home/mocchiut/pamela/PamCalo/lib/Linux/libPamCalo.so")
root [1] TFile *_file0 = TFile::Open("pamelasimu.root")
root [2] _file0->ls()
TFile**         pamelasimu.root
 TFile*         pamelasimu.root
  KEY: TTree    pamcalotree;1   PAMELA simulations
root [3] pamcalotree->Show(0)
======> EVENT:0
 PamCalo         = (PamCalo*)0x9974f98
 fUniqueID       = 0
 fBits           = 50331648
 energy          = 14.856709
 pID             = 0
 qtot            = 55.280846
 qmax            = 3.763029
 qtrack          = 55.280846
 qcore           = 289.221985
 qcyl            = 55.280846
 qlast           = 10.961176
 qpre            = 7.893903
 qtr             = 55.280846
 q0              = 0.982600
 q1              = 0.000000
 q2              = 1.932100
 ... ... ...
```

Exercise 1

Draw variables versus energy (scatter plots), try:

```
qtot/abs(energy):energy
noint:energy
qcore/ncore:energy
qtrack/qtot:energy
```

(that is:

```
root [] pamcalotree->Draw("qtot/abs(energy):energy")
```

etc. etc.)

Look at differences between negative and positive sides of the distributions (next lesson: where are the positrons?).

Exercise 2

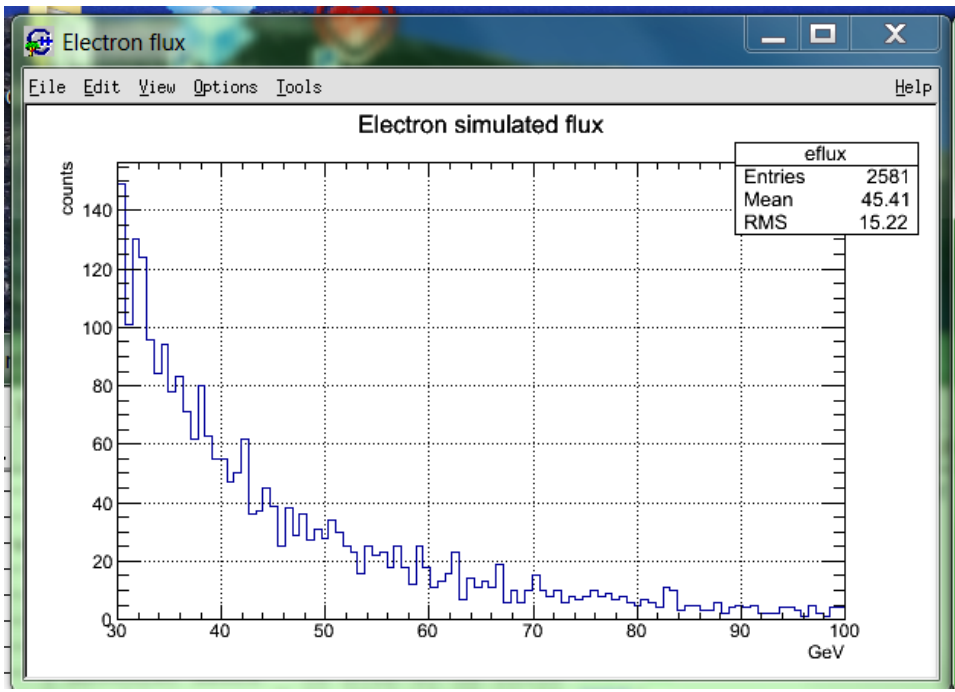
```
void readClassTree(TString inputFile){

  TCanvas *c1 = new TCanvas("c1","Electron flux",200,10,600,400);
  c1->SetGrid();
  c1->SetTicks();
  c1->Draw();
  TCanvas *c2 = new TCanvas("c2","Energy momentum match",200,10,600,400);
  c2->SetGrid();
  c2->SetTicks();
  c2->Draw();

  // Create some histograms.
  TH1D *eflux = new TH1D("eflux","Electron simulated flux;GeV;counts",100,30.,100.);
  TH2D *qnstrip = new TH2D("qnstrip","qtot/nstrip;GeV;qtot/nstrip",1000,-100.,100.,1000,0.,35.);

  // Open the file
  TFile *file = TFile::Open(inputFile);
  TTree *tree = (TTree*)file->Get("pamcalotree");
  PamCalo *pc = new PamCalo();

  tree->SetBranchAddress("PamCalo",&pc);

  for ( Int_t i=0; i<tree->GetEntries(); i++) {

    tree->GetEntry(i);

    if ( pc->energy > -100. && pc->energy < - 30. ) eflux->Fill(fabs(pc->energy));
    qnstrip->Fill(pc->energy,pc->qtot/(float)pc->nstrip);

  }

  c1->cd();
  eflux->Draw();

  c2->cd();
  c2->SetLogz();
  qnstrip->Draw("colz");

}
```

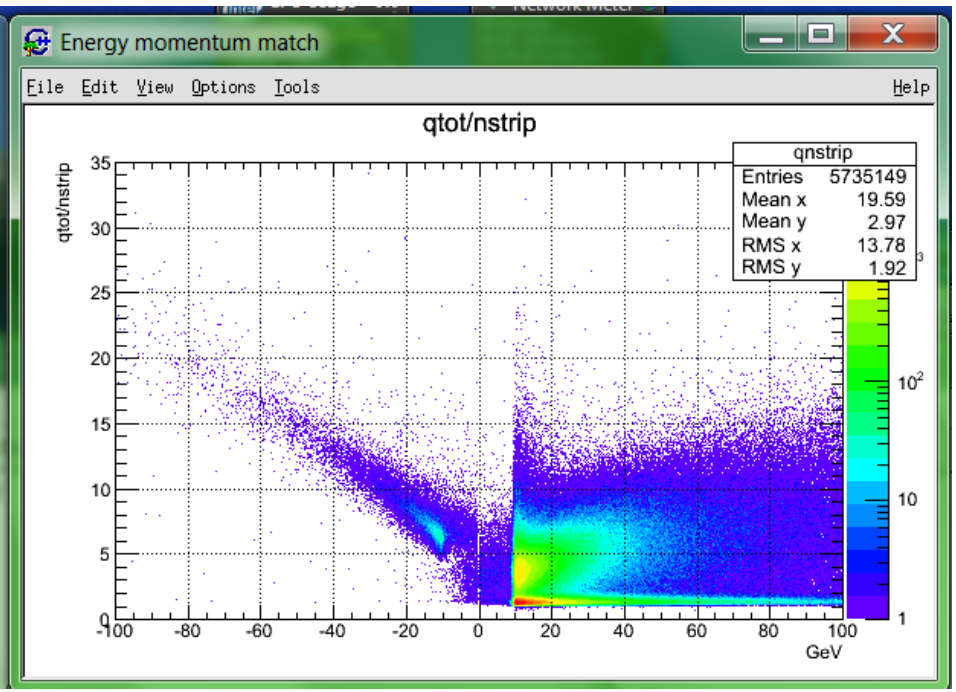
Script readClassTree.C (download it from moodle)

Exercise 2

Run it like this:

```
bash> root
root [0] .L readClassTree.C
root [1] readClassTree("pamelasimu.root")
```





Exercise 2

Change readClassTree.C in order to:

- fit the electron flux distribution with a power law

```
TF1 *plaw = new TF1("plaw","[0]*pow(x,[1])");
```

in the energy range [5,20]; which value do you get for the parameter number 1?

- plot into a TH1D (x axis range: 0., 200.) the variable "qtot" in the energy range -20 < energy < 0 ; fit the resulting histogram with a gaussian function, print (cout from the script) the resulting mean; which value do you get?

Exercise 3

1. Download from moodle the script `posSelection.C`
2. Run the script and look at the output plots.
3. Add new plots and selections in order to throw away protons, try to use:
   - qtrack/qtot
   - qmax/qtrack
   - qlast/qpre
   - qpre/npre
   - qlast/nlast

   versus energy. Take qtot/energy distribution as a reference.

   What do you get as positron fraction? is it what you expected?

   At this point, does the qtot/energy distribution bring additional information to reject protons?

Exercise 4

4. take the last version of your script and create two new variables using the variables: q0, q1, q2, q3, q4, q5. Event by event basis:
   - create a TGraphError using as x vector x[6]={0.,1.,2.,3.,4.,5.}, as y vector the six q*N* values and as errors the square root of q*N;*
   - fit the resulting TGraphError object with an exponential function:

     $$f(x) = A \exp(Bx)$$

   - plot the resulting values of A (range about [0.-10.]) and B (range about [-2., 2.]) versus energy into two different scatter plots, do they give an additional information to separate protons and positrons?